中华学习机实用大全①

BASIC与Logo语言

韩仲清 主编



電子工業出版社

中华学习机实用大全①

BASIC 与 LOGO 语言

韩仲清　主编

电子工业出版社

内 容 提 要

本书从实用的角度出发，详细介绍了中华学习机 CEC-I 的 BASIC 与 LOGO 语言。主要内容包括：

中华学习机的主要部件及其安装；BASIC 语言的基本概念；简单的 BASIC 程序；转向和分支程序；循环程序；数组和函数；子程序；顺序文件和随机文件的建立、检索以及修改方法；LOGO 语言的基本概念；海龟直接作图和程序作图方法；数、字和表处理以及函数曲线的绘图方法和技巧。

本书的最大特点是内容丰富、具体、实用、易学。最适宜于广大青少年、中小学生及其家长、各类办公人员、企业管理人员和计算机爱好者自学；也可以作高等院校非计算机专业、培训班、函授班、职业学校、中专和中小学等学生计算机课程的教材；还可供从事计算机研究和应用的人员使用。

中华学习机实用大全①

BASIC 与 LOGO 语言

韩仲清 主编

责任编辑 崔囤荣 王昌铭

电子工业出版社出版（北京市万寿路）

电子工业出版社照排室排版

电子工业出版社发行　各地新华书店经售

中国科学院印刷厂印刷

开本：787×1092 毫米 1/32 印张：11.25 字数：259 千字

1990 年 9 月第 1 版　1991 年 5 月第 2 次印刷

印数：14200—24300 册　定价：4.40 元

ISBN 7-5053-1046-1／TP · 169

前　　言

中华学习机以前所未有的速度进入寻常人家，成为人们工作、学习和生活的得力助手，尤其是在开发青少年的智力方面，已经显示出了强大的威力。

为满足广大青少年、中小学生及其家长和计算机爱好者对中华学习机知识的渴求，我们组撰了这套《中华学习机实用大全》。该书内容丰富、具体、实用；把中华学习机的最新软件以及最实用、最急需的技术、技巧和方法毫无保留地介绍给读者，使初学者很快入门，入门者进一步提高；学到知识，掌握技术，增长才干，启迪智慧，得到力量，增强解决实际问题的能力。

《中华学习机实用大全》分为七册：

1. BASIC 与 LOGO 语言
2. 汉字处理与数据库技术
3. 操作系统
4. FORTRAN 与 Pascal 语言
5. 汇编语言程序设计
6. 游戏与绘图
7. 硬件维修与经验技巧

为便于阅读和使用，每册内容彼此均是独立的，读者可以从任何一本书开始阅读。但是，如果读者是计算机技术的初学者，那么最好按顺序阅读，当然，每本书中可以只选学自己感兴趣的那部分内容。

《中华学习机实用大全》在内容安排上，由浅入深，循序渐进。既考虑到初学者很快入门，又考虑到让入门者进一步提高，

还考虑了应用者能够实用。书中有较多实例，读者可以边读、边学、边用、边想、边写(写自己的程序)。在结构安排上，既便于自学，又可以作为教材。在文字叙述上，力求浅显、通俗、易懂。在选材上，突出实用性技术。

本书是《中华学习机实用大全》的第一册，主要介绍了中华学习机的主要部件及其安装；BASIC 语言及其程序设计方法；LOGO 语言程序设计。其中介绍的利用 LOGO 语言来绘制函数图形是同类书中所没有的内容。

欢迎读者对这本书进行品评，指出疏漏和错误，我们将甚为感谢！

在编写本书的过程中，电子工业出版社和电子报社的编辑们给予了指导和帮助，特别是王有春、孙萌老师提出了许多宝贵的修改意见；为调试和运行示例程序，成都三开元电脑部经理舒新生无偿地提供了 CEC-I 中华学习机及其软件；张升楷副教授审阅了全部书稿。在此一并表示感谢！

参加本书编写的有：韩仲清，车陆翔，廖兴祥，刘元社，揭金良，卿文模。全书由韩仲清统稿。

编者

1989 年 11 月 14 日于四川大学

目　　录

第一部分　中华学习机及 BASIC 语言程序设计

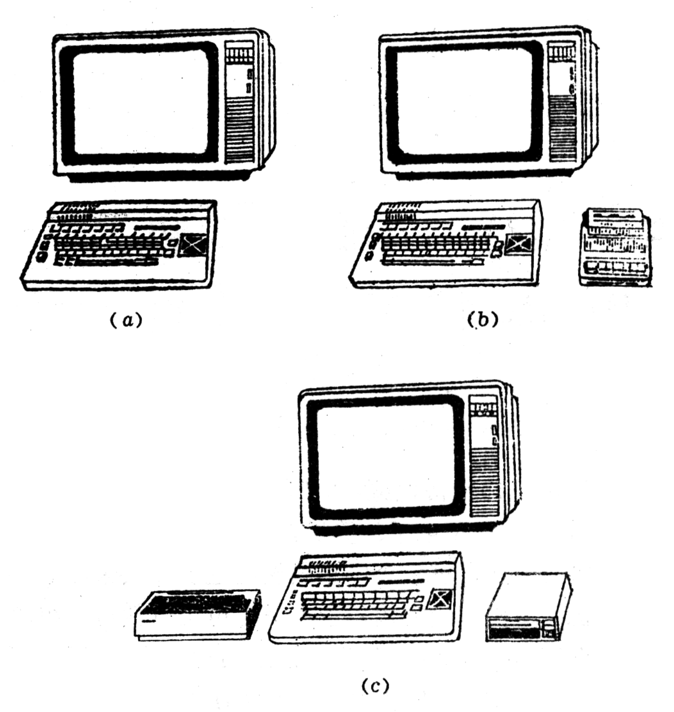
第二部分　LOGO语言程序设计

第一部分　中华学习机及 BASIC 语言程序设计

中华学习机是一种价格低廉，功能较强，结构灵活，使用简便，无特殊环境要求的微型电子计算机。该机迅速普及到中、小学，进入家庭，将有力地促进整个中华民族的进步和繁荣。

中华学习机的主机配上家用彩色或黑白电视机就可以组成最基本的微机系统。如图Ⅰ(a)。如果增配家用录音机(可以是廉价的盒式录音机)就构成了在录音磁带上保存程序和数据的中华学习机系统。如图Ⅰ(b)。再配上打印机及软盘驱动器就构成一台和 APPLE Ⅱ(苹果)机完全兼容，功能相当且在汉字处理能力方面有所增强的能用于中、小型计算与事务处理的计算机系统。如图Ⅰ(c)。这样的中华学习机系统其价格约为 APPLE Ⅱ 机之一半。

这部分除了介绍中华学习机的硬件安装以外，还着重介绍了 CEC-I 中华学习机上 BASIC 语言的程序设计方法和技巧。



(a) (b) (c)

图 I　三种配置的学习机系统

第一章　中华学习机的主要部件及其安装

1.1　主要部件及其作用

1.1.1　主机

中华学习机的主机在机壳内（键盘下面）的主电路板上。它由中央处理器 6502、存储器电路、输入输出接口电路和电源所组成。

中央处理器 6502 是一片大规模集成电路。它的时钟为 1MHz，一种为＋5V 的电源。有 8 条数据总线，16 条地址总线（寻址范围 64K 字节）。中央处理器 6502 是整个机器的核心。

存储器电路可分为半导体存储器和存储器管理部件。半导体存储器有 RAM 和 ROM 两种。由两片 64K×4 位的集成电路 50464 组成 64K 字节的 RAM。由一片集成电路 27256 组成 32K 字节的 ROM。ROM 上固化有监控程序及 BASIC 语言。存储器管理部件由门阵列器件组成。在主电路板上，共使用了三片专用集成电路，除存储器管理部件外，还有一片门阵列器件为输入输出管理部件和一片可编程阵列逻辑电路 PAL，其作用是产生时序信号及存储器行列地址选通信号等。

一般在主电路板上的扩充汉字系统插座里，插有一片固化

有汉字系统的 ROM 片子 27256，两片固化有汉字字库的 ROM 片子和两片 TTL 电路。

1.1.2　键盘

中华学习机的键盘安装在主机外壳上，具备有标准打字机键盘的各种键，键位的排列同标准打字机。见图 1.1。

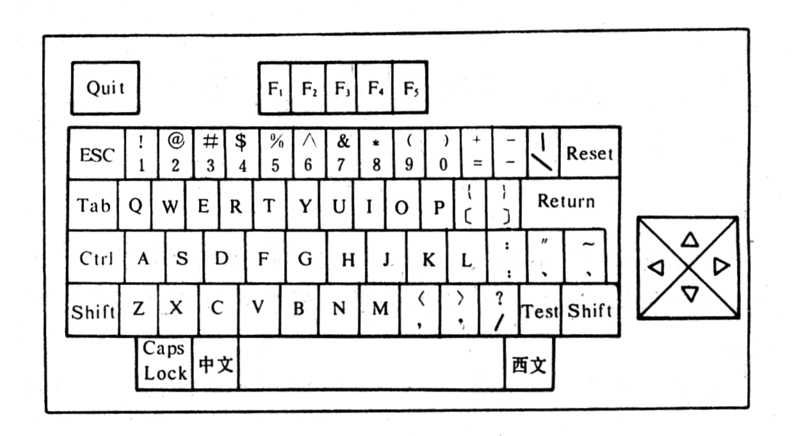


图 1.1　键盘

键盘和主机通过一条 26 线扁平电缆连接，直接装入主机壳内。用户通过键盘向主机输送信息。此键盘具有重复功能和 N 键滚动功能。

所谓重复功能是指当某键按下的时间大于一定值时，则连续不断地重复发送这个键值，并被主机接受。

所谓 N 键滚动是指在有多个键按下时，则根据被敲入键的先后次序分别检测并输入主机，同时也显示在屏幕上。

键盘的技术性能：

键数：69 个

字符编码方式：ASCII 码

编码个数：128 个

接口器件：KB3600-PRO

特殊键功能说明：

Capslock：大小写字母转换键，又称锁定键，当按下此键，再敲字母键时，则字母全是大写。若又按一次锁定键，即抬起锁定键，则再敲的 26 个英文字母键都是小写。

Shift：当按住此键后再敲另一键时，按键上半部标注的字符输入主机并显示在屏幕上。

Esc：按此键后，再敲其它 ASCII 键产生新的 ASCII 码输入主机，一般将此键和◁，▷，△，▽键用于屏幕编辑。

Ctrl：这是一个与其它键同时按下产生某种控制码的键，例如：在运行 BASIC 程序时，同时按下 Ctrl 和字母 C 两个键，则中断正在运行的 BASIC 程序，回到 BASIC 提示状态。

Tab：屏幕定位键。如屏幕上已定了 11、21、31、41 四个位置，光标在 13 那儿，敲一下 Tab 键就移到下一个定位 21。

Retrun：回车键，程序中通常用↙表示，用于结束一个命令，或者是换行，功能同打字机上的 CR 键。

Reset：同时按下本键与 Ctrl 键，使机器系统复位（或称重置系统，即重新启动系统）。

▷：敲此键使屏幕上的光标右移，并把它经过的那些符号如同从键盘输入一样送入主机。

◁：敲此键使屏幕上的光标左移，并把它经过的那些符号从主机中删掉。

△：光标上移。

▽：光标下移。

Test：当同时按下 Ctrl-Reset-Test 三个键时，主机进行自检。

F1:进入 ASCII 方式。按此键后,再敲入的键主机将按 ASCII 码接收并显示之。

F2:进入拼音方式。按此键后,即可用对应的拼音字母输入汉字。

F3:进入区位码方式。按此键后,即可用区位码输入汉字。

F4:计算机系统把该键留给用户作为自定义的功能键。键码为:14。

F5:留给用户作为自定义的功能键。键码为:06。

Quit:按此键使系统从监控状态返回到 BASIC 状态,此键不破坏内存里原有的 BASIC 程序。该键的功能相当于同时按 Ctrl-C 两个键。

1.1.3 显示器

显示器是计算机最基本的输出设备,是人一机对话的主要工具之一。它不仅显示用户输入的字符,以便即时看到已输入的字符是否正确,而且能显示主机运行的过程及结果,并且还会显示主机对用户所提出的要求和命令。

中华学习机的显示器可以用通常家用的彩色或黑白电视机(PAL-D 制式)代替,也可以用单色或彩色(PAL-D 制式)监视器。

对于彩色或黑白电视机,只要是 PAL-D 制式的,即能接收我国电视节目的电视机,均能用作中华学习机的显示器。

中华学习机的显示方式有五种。

1. 字符显示

能显示 5×7 点阵构成的 96 种字符(西文字符)。显示屏上共有 24 行,每行 40 个字符,即可显示 960 个字符,可以显示正相的(黑底白字),也可以显示反相的(白底黑字),或闪烁显示

(即上述两种情形的交替)。

2. 低分辨图象显示

可显示 48×40(1920)个彩色小方块。每个小方块可在 16 种不同颜色中选择。

3. 高分辨图象显示

可显示 192×280 个彩色点。每个点可在 8 种不同颜色中选择。

4. 混合显示

可以是 1、2 两种方式的混合,也可以是 1、3 两种方式的混合。各种方式的转换均由程序通过软开关实现。

5. 中文方式

中文显示采用前述高分辨图象显示方式,以 16×16 的点阵显示构成 7445 种图形字符,其中包括 6763 个汉字及英文字母、日文假名、拉丁字母、俄文字母、希腊字母等。显示屏上共能显示 11 行。前 10 行,每行显示 17 个汉字,共可以显示汉字 170 个。第 11 行用来显示当前的输入方式和相应的提示信息。

1.1.4 打印机

中华学习机可通过打印机接口板与打印机连接。打印机接口卡可插入主机扩充槽与主机 50 线总线相连。中华学习机可配 CP-80、RX-80、MX-80、FX-80 或 MX-100、FX-100 等多种型号的打印机。在汉字管理程序中,打印机驱动程序的设计仅适用于九针图形打印机。其功能要求必须与 EPSON MX-80Ⅲ型打印机兼容,如:FX-80Ⅲ、RX-80Ⅲ、CP-80Ⅲ、FX-100+ 和 YAMATO 等型号的打印机。对于 MX-80Ⅱ、FX-80Ⅱ、RX-80Ⅱ可以用修改控制命令的方法来解决。

有了相适宜的打印机就可以把用户的文件、程序、报表、图

形等打印出来，作为资料保存使用。

1.1.5 软盘和软盘驱动器

软盘的作用是保存程序、数据等资料信息，它与唱片相似。软盘驱动器是实现对软磁盘进行读、写操作的硬设备。中华学习机要求使用 5.25 英寸单面单密度，容 量为 143K 字节的软磁盘，见图 1.2。

1. 软盘片使用注意事项

①软盘片不能受重压，不可弯曲，使用后要及时插入盘片的纸盘套内，立放于软盘盒中。

②使用时将软盘片有标签的一面朝上，用右手拇指按住有商标的那一面，将有磁头读/写窗口的一边先插入驱动器，然后慢慢地试探性地推入软盘片，直到盘片全部进入驱动器，再关好驱动器的门。

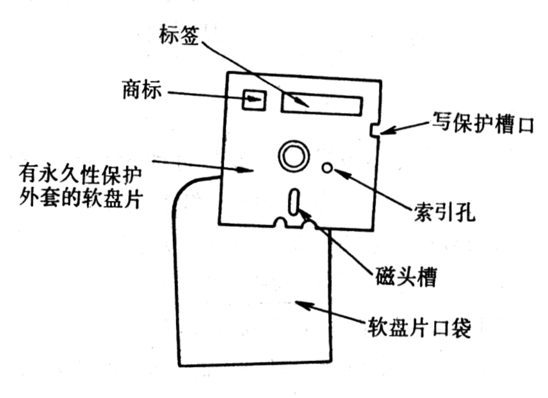


图 1.2 软盘

③取出时用食指和拇指捏住盘片的边缘，反方向抽出。

④不能在驱动器旋转和亮灯时，打开驱动器门，抽出或插入

软盘片。如此容易损坏软盘片，丢失盘片上的信息。

⑤使用中要保持盘片的清洁，避免灰尘和油污的污染，绝对禁止用手触摸暴露在磁头读/写窗口和索引孔中的磁盘部分。

⑥盘片的保存温度最好在 10℃～50℃之间，避开磁场的干扰，保持干燥。

2. 软盘驱动器使用注意事项

①对软盘驱动器要做到轻拿轻放，避免剧烈震动，以防止定位机构松动。

②避免在软盘驱动器工作时移动盘驱本身。

③禁止在开机状态下插或拔软盘驱动器和软盘驱动器接口卡。

④不可在马达转动时抽、插软盘片。

⑤温升变化范围应小于每小时 15℃，环境温度应在 10℃～38℃之间。

⑥避免灰尘侵袭，不要将沾有灰尘的软盘片插入到软盘驱动器中。

⑦磁头可用专用的清洗软盘片或磁带的清洗液。

⑧软盘驱动器应远离磁场发生源，不用时不要将软盘片留在驱动器中。

1.1.6 录音机

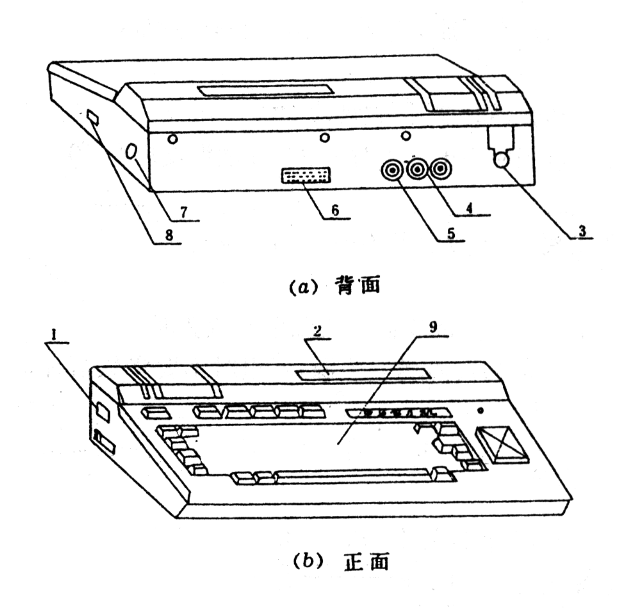
录音机是一种廉价的外部存储器，由于一旦关掉电源，主机内存中的内容就会消失，所以除了用软盘外，还可以用录音机磁带保存程序、数据等资料信息。中华学习机要求录音机具有录放功能并带有 EAR 和 MIC3.5mm 插孔。建议使用工厂推荐的配套盒式录音机。

1.1.7 电源

在主机壳内左边有一个开关电源。输入电压为220V,交流50Hz,最大功率小于25W。由开关电源输出±5V,±12V供主机及磁盘驱动器接口工作。

中华学习机可以不需要专门的电源稳压器。

1.2 部件之间的线路连接



(a) 背面

(b) 正面

图1.3 主机接口图

1.电源开关;2.扩充槽口;3.电源线;4.电视机接口;5.监视器接口;6.磁盘驱动器接口;7.录音机接口;8.游戏杆接口;9.键盘

首先从中华学习机主机箱中小心取出主机,并清点随机带

的视频电缆，录音机电缆，使用手册等附件。

主机右上角有一个电源指示灯（小孔内）。电源一旦接通，此灯便发亮。前半部是键盘，机壳侧面装有各种接插件与外部设备连接。

主机上的接口请参看图 1.3。

提醒用户注意：如果读者是初次安装计算机系统，各部件线路连接之后，请不要急于打开电源开关，仔细检查各部件之间的线路连接确认正确无误后，再开启电源，如有老师指导，最好请老师检查后，再开电源。

1.2.1 主机和屏幕的连接

屏幕可以用电视机（彩色或黑白）或监视器（彩色或黑白），但连接的方法不一样。

1. 电视机的连接

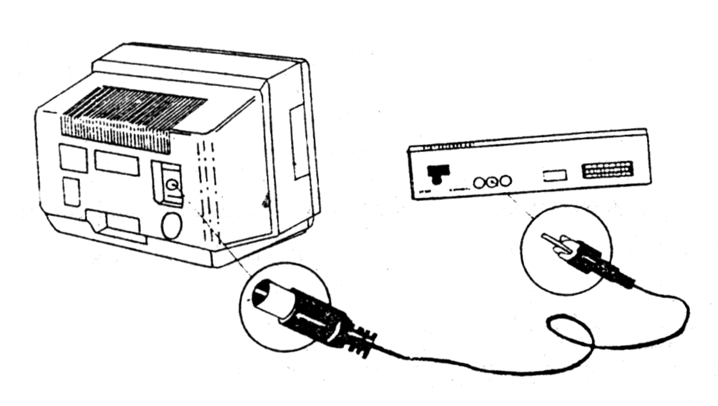


图 1.4 电视机的连接

第一步：将主机和电视机放在桌上，并配备两个电源插座。

第二步：把电视机的天线插头拔去。

第三步:用视频电缆线一头插入主机的电视机接口,一头插入电视机天线插座,要求电视机天线插座的输入阻抗为 75Ω。见图 1.4。

第四步:确认主机电源开关处于“OFF”(关闭)位置。见图 1.5。

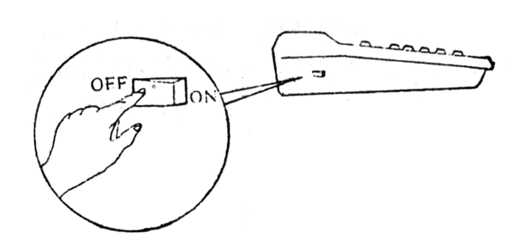


图 1.5 主机电源开关位置

第五步:将电视机电源插头和主机电源插头分别插入电源插座。

第六步:把电视机的音量开关拧到最小位置。

第七步:先开电视机再开主机。

第八步:当使用黑白电视机时,首先调准电视机频道(以出厂厂家要求为准)。调节频道微调及对比度、亮度旋纽,直到屏幕出现清晰的“ZHONG HUA XUE XI JI”字样为止。

当使用彩色电视机时,同样先选好一个频道,并通过自检方式进入显示彩条图象,然后调节频道微调及饱和度、亮度、对比度旋纽,直到屏幕上的彩条满意为止。以后可把此频道作为连接计算机的专用频道,再开机时就不用调机了。

注:主机的电视机接口输出阻抗是 75Ω。

2. 监视器的连接

第一步:事先准备好两个电源插座,把主机和监视器放于桌上。

第二步:用视频电缆线一头插入主机的监视器接口,另一头插入监视器 IN 插座,见图 1.6。

第三步:确认主机电源为“OFF”(关闭)位置。

第四步:把主机电源及监视器电源的插头分别插入电源插座。

第五步:先开监视器后开主机,屏幕将出现清晰的“ZHONG HUA XUE XI JI”字样。

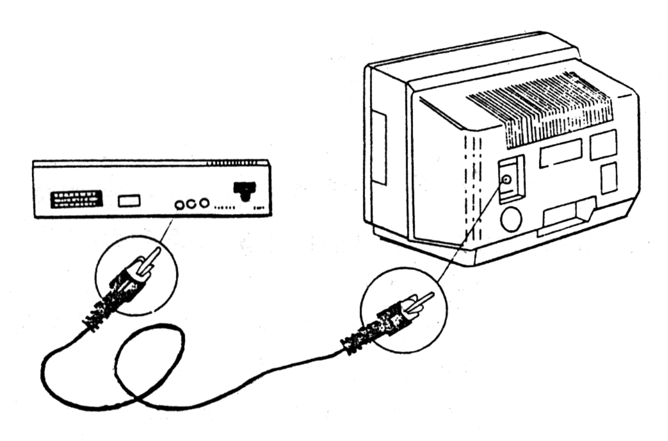


图 1.6 监视器的连接

1.2.2 主机和录音机的连接

第一步:确认主机电源为“OFF”位置,再备一个电源插座(除主机、显示器电源插座外)。

第二步:把录音机电缆线的一端(五芯插头)插入主机侧面的录音机接口,必须注意:五芯插头的凹口部分在正上方。

第三步:电缆线的另一端的两个 3.5mm 插头,分别插入录音机 EAR 插孔(连接主机 IN)和 MIC 插孔(连接主机 OUT)。

第四步:把录音机电源线插入电源插座。参见图 1.7。

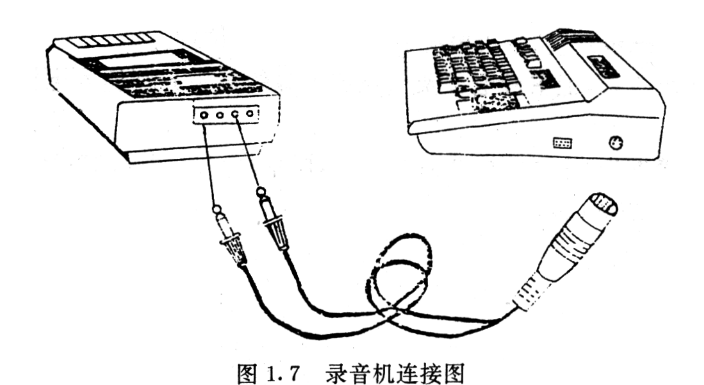


图 1.7　录音机连接图

1.2.3　主机和扩充卡的连接

主机的右上方有个 50 线的扩充槽,可插多种外设卡和扩充卡(如打印机卡,音乐卡,A/D 转换卡等)。扩充卡尺寸太大插不进去,可采用 50 线转接板把槽口抬高,再插外设卡或扩充卡。

第一步:确认主机电源为"OFF"位置。

第二步:用手轻轻压下扩充槽盖,可看到印制板上安装了一个 50 线插槽。

第三步:把卡按照元件面朝前,焊接面朝后的方向对准槽口往下插。必须注意:一定要平插插平,以免错位。

扩充槽信号排列参看图 1.8。

扩充槽的槽号,用户可以用短接帽来选择,槽号可以为 1,2,4,5,7 中的任一个。3 号槽已用于汉字系统,6 号槽已固定用于磁盘驱动器接口,故不可再选用它们。出厂时一般都设在第 1 槽,即将两个短接帽插在标有"1"的位置上。选择槽号插头位于主机板的右侧,并标有槽号。如需改槽号,需打开主机壳,把原在

1号槽位置的两个短接帽取下并插在所需要的槽号插头上。

1.2.4 主机和打印机的连接

用 1.2.3 节介绍的方法将主机和打印机卡连接好后，再用打印机与主机的 20 线（扁平）连接电缆连接主机与打印机。

第一步：确认主机及打印机电源开关为“OFF”位置。

第二步：将连接主机接口卡（小）的一端及连接打印机接口（大）的一端，分别插合在主机上的打印机卡接口及打印机的接口上（注意两端接触良好）。

第三步：先开打印机后再开主机电源开关。

1.2.5 主机和磁盘驱动器的连接

第一步：确认主机电源开关处于“OFF”位置。

第二步：把磁盘驱动器电缆线（20 线扁平电源）的一端连接驱动器，另一端插入主机上的驱动器接口。必须注意：插针与插孔必须一一对应，电缆线两端的凸出部分都朝

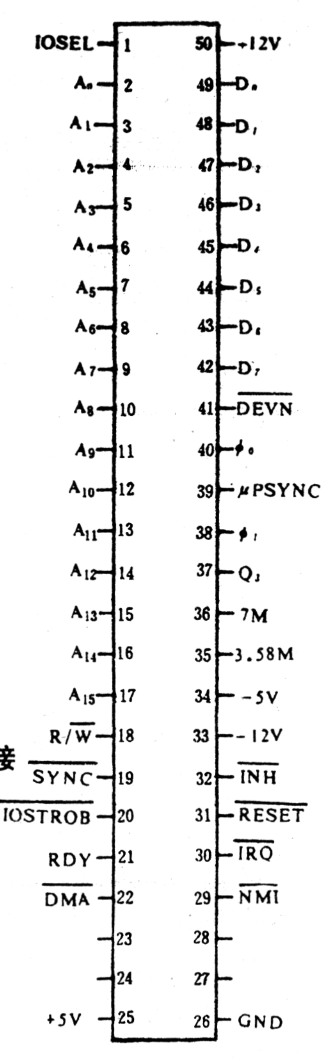


图 1.8 扩充槽信号排列图

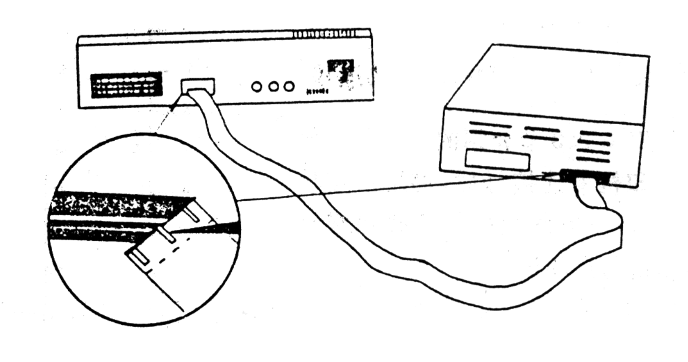


图 1.9 驱动器的连接

上。参见图 1.9。

1.3 中华学习机的自动检测

同时键入 Ctrl-Reset-Test 三个键且最后放开 Test 键，可进行主机检测，亦称主机自检。即对 RAM、ROM 进行检测，显示器显示相应的检测内容和结果。如图 1.10。

其中 RAM 部分是测试 $0000～$BFFF 48K RAM。

BNK1、BNK2 部分是用两个软开关测试 $D000～$FFFF RAM。

当测试正确，屏幕显示 OK，如有错，屏幕相应处显示 ERR。

ROM1 是测试 LOGO 语言的。

ROM2 是测试中华学习机 CEC-BASIC 和监控内容的，ROM1 和 ROM2 是合在一块 27256ROM 片上的。

AUX1 测试汉字码表内容。

AUX2 测试汉字管理系统内容。AUX1 和 AUX2 合在一块 27256ROM 片上。

AUX3 测试两块 1 兆位的汉字库内容。

MEMORY — TEST

TIMES:0001

RAM	BFFFF	OK
BNK1	FFFF	OK
BNK2	FFFF	OK
ROM1	BFFF	LOGO
ROM2	FFFF	CEC-BASIC
AUX1	BFFF	HZTABLE
AUX2	FFFF	HZPROGRAM
AUX3	7FFF	CECWL

图 1.10 主机自检内容

测试正确，屏幕将显示相应的测试内容名称，如果测试出不为所知的内容，屏幕将显示 UNKNOWN（不认识）。当没有加汉字库时，屏幕将在相应的位置显示 NULL。

当 RAM、ROM 测试完毕，程序进行彩色显示测试，屏幕将显示出 16 种不同颜色的彩条。

自检程序可连续循环地测试，并在 TIMES：一栏显示测试的次数。如需调试彩电，可在屏幕显示彩条时敲任何一个键使屏幕

一直显示彩条，直到您调试好彩电，再按任一个键又进入自检程序循环测试。如要中断自检，请同时按下 Ctrl-Reset 键，系统便退出检测而进入 BASIC 状态。

1.4 中华学习机的主要技术指标

CEC-Ⅰ中华学习机的主要技术指标、性能如下：

1. 主机

①中央处理器：6502，8 位微处理器。

②内存容量：64K 字节 RAM。

③32K 字节 ROM，固化监控程序及 BASIC、LOGO(子集)语言。

④专用集成电路：一片门阵列器件 MMU 为存储器管理部件，一片门阵列器件 IOU 为输入/输出管理部件，一片可编程阵列逻辑电路 PAL 产生时序信号。

⑤汉字系统：汉字系统已做在主机板上，可根据用户需要取舍。该系统提供拼音、区位、音形等输入方式，采用全点阵的国标一、二级汉字字库(两片一兆位 ROM)，包括 6763 个汉字及外文字母等。

⑥显示器接口：一个是经过调制器输出的射频信号，可连接 PAL 制式彩色或黑白电视机；另一个是全电视视频信号(没有音频)输出，可连接标准的 PAL 制式彩色或单色监视器。

⑦盒式磁带机接口：该接口可与一般家用录音机连接，作为廉价外存设备。提供输出电压是 25mV，输出阻抗 100Ω，要求输入信号峰-峰值电压是 1V，输入阻抗 12kΩ。

⑧磁盘(软盘)驱动器接口：磁盘驱动器接口已做在主机板上，用于连接一个 5.25 英寸的磁盘驱动器。

⑨游戏控制杆接口：九针的游戏杆插座提供三个 TTL 电平开关量输入，四个模拟量输入。

⑩扬声器接口：内接 0.25W 8Ω 扬声器。

⑪键盘：键盘是标准的打字机键盘，具有 69 个键，包括大小写字母、数字及一些特殊功能控制键。采用弹簧键体，键盘通过 26 线扁平电缆直接连接主机板的键盘接口，装在主机上。

⑫扩充槽口：在主机板上留有一个与 APPLE Ⅱ 系列兼容的 50 线输入/输出插座。此插座可插打印机接口电路或其它扩充电路板。

⑬电源：采用开关电源，提供四路电源，额定负载为：+5V、1.7A；-5V、0.1A；+12V、0.8A；-12V、0.1A。总功耗约 25W。

2.　显示器

可用彩色或黑白电视机，也可用彩色或单色监视器，其屏幕显示功能为：

①西文方式

字符构成：5×7 点阵。

每帧字符：40 字符×24 行。

显示方式：正常，反相，闪烁。

②中文方式

字符构成：16×16 点阵(汉字)，8×16 点阵(ASCII 码)。

每帧字符：17 字符(汉字)×10 行，

　　　　　34 字符(ASCII 码)×10 行。

另加一个状态行。

③图形方式

低分辨率彩色图形显示：分辨率为 40H×48V，16 种颜色。

高分辨率彩色图形显示：分辨率为 280H×192V，8 种颜色。

3.　软盘驱动器

5.25 英寸软盘驱动器，采用单面单密度记录方式，格式化后容量为 143K 字节。

4. 盒式磁带机

任选专用磁带机或普通盒式录音机，用普通标准录音磁带。

5. 打印机

打印机与可扩充的打印机接口电路板相连。Centronic 标准接口可配多种点阵打印机和彩色绘图打印机。

6. 系统软件

包括监控程序、DOS3.3、CEC-BASIC、LOGO（子集）、汉字管理软件、CdBASE Ⅱ、WORDSTAR、CP/M 操作系统、FORTRAN 语言、Pascal 语言、6502 汇编语言等。

第二章　BASIC 语言的基本概念

从本章起，我们针对中华学习机 CEC-Ⅰ，开始介绍 BASIC 语言的一些基本概念和程序设计方法。这些概念和方法不仅适用于中华学习机，而且也适用于一般计算机的 BASIC 语言。掌握了 BASIC 语言后再去学习其它高级语言也就不困难了。

2.1　什么叫 BASIC 语言

2.1.1　BASIC 是什么意思

对于 BASIC 语言，读者可能并不陌生。因为很多场合都能碰到它。但究竟是什么意思，有些读者，特别是计算机技术的初学者可能就不太清楚了。单就 BASIC 这几个字符来说，它是英文 Beginer' s All-purpose Symbolic Instruction code 的缩写，意思是初学者的通用符号指令代码。它代表了一种语言。但这种语言不同于日常生活中用于交流思想感情的自然语言，它是一种用于描述小型数值计算问题和事务管理的计算机算法语言。

就一般来说，BASIC 语言总是与某类微电脑联系在一起。如中华学习机 CEC-Ⅰ，它使用的 BASIC 语言通常说成 CEC-BASIC，表示 BASIC 语言适用于具体的 CEC-Ⅰ机。类似的有 APPLE Ⅱ BASIC，紫金 Ⅱ BASIC，IBM PC BASIC 等等。

对于初学者来说，只要掌握了某种计算机上 BASIC 语言的使用方法，对于其它计算机上的 BASIC 语言也就容易触类旁通

了。因为从总体语法规则来说，不同计算机上的 BASIC 语言总是大同小异。因此，建议读者在学习一种语言（不论是什么语言）时，先结合一台具体的计算机，边学边用，学用结合，这样可以收到多快好省的效果。

2.1.2 BASIC 语言的特点

BASIC 语言自身有很多特点，但归纳起来，最重要的特点有：

1. 语法规则比较简单、直观，容易为大多数人理解和接受。因而容易学会使用。

2. 程序结构简便。由于大多数微机上的 BASIC 语言都采用解释性的方法来实现，因而对 BASIC 语句的增加删改都比较容易。

3. BASIC 语言具有很强的人一机会话功能。人们在使用时，能够随时掌握语句的执行情况，能够及时调整和交换信息。

由于 BASIC 语言具有简便实用的特点，因而目前几乎所有的微机上都配备了 BASIC 语言系统。随着计算机技术的不断发展，BASIC 语言自身 也在不断改进和提高。总的发展趋势是：功能越来越强，使用越来越方便。我们着重介绍的中华学习机上的 BASIC 语言，已经固化在主机的只读存储器中，称 CEC- I BASIC，简称 BASIC，其功能相当于扩展 BASIC。它除了具有 APPLESOFT BASIC 的功能外，还提供了对汉字的输入、显示和打印等功能。可以说，CEC-BASIC 是一个小型多功能语言系统。

2.1.3 BASIC 语言的使用方式

BASIC 语言有两种使用方式（亦称执行方式）。一是人一机对话方式。所谓人一机对话方式，就是指在机器上敲入一个命令

后机器就立即执行之，其执行结果立刻在屏幕上显示出来。这种一问一答的方式，最适宜于初学者作为熟悉 BASIC 语言练习之用，因为看得见，很实在。另一种方式就是程序设计方式。这种方式的特点是将需要进行的计算或事务管理工作编制为统一的程序，再输入机内一起执行。这种方式速度快，程序能够自动执行，适宜于实际应用。当然，两种方式在使用中并不能截然分开。在本书中，我们着重介绍程序设计的工作方式。

下面用一个例子来说明两种执行方式的主要区别。

例 2.1　计算梯形的面积。公式为

S=(上底+下底)×高/2。

假设用 A 表示上底，B 表示下底，H 表示高。给出一组数据为：3，4，5(即上底=3，下底=4，高=5)。

用对话方式，BASIC 执行情况如下(↙表示回车键)：

```
]A=3 ↙                    将上底值 3 送到 A
]B=4 ↙                    下底值 4 送到 B
]H=5 ↙                    高的值 5 送到 H
]S=(A+B)*H/2 ↙            计算梯形面积
]PRINT  S ↙               打印计算结果
17.5                      运算的结果
```

用程序设计方式如下：

```
]10 INPUT  A,B,H ↙        使 A,B,H 得到值
]20 S=(A+B)*H/2 ↙         计算面积
]30 PRINT  S ↙            打印计算结果
]RUN ↙                    运行程序
? 3,4,5 ↙                 输入数据
17.5                      输出运算结果
]                         回到 BASIC 状态
```

如果读者亲自上机一试，就会发觉：对话方式的特点是敲入

一个语句后就立刻执行那个语句，并且语句之前不带行号（即每行前面给出的无符号整数）。而程序设计方式是把所有语句全部输入机内，最后发布运行命令 RUN，机器才从行号小的语句开始执行，输出运行结果。如果对上述语句中的符号不清楚，没关系，待下一章介绍后就自然明白了。

2.1.4 本部分用到的书写记号约定

为了叙述的方便，本部分经常用到以下一些记号。这些记号不是 BASIC 语言自身的内容，引入这些记号纯粹是为了说起来简便。

[…]——表示方括号中的内容可根据实际需要进行取舍，称为待选项。

〈…〉——表示尖括号中的项在实际书写程序时必须填入具体的内容，但程序中并不书写尖括号。

↙——表示回车键，即键盘上的[Return]键。

[Ctrl]+[Reset]——表示先按住[Ctrl]键不松手，再敲[Reset]键。（其它语言描述中，也有类似的约定）。

2.2 BASIC 语言用到的符号

任何一种语言，包括自然语言在内，都要用到一些符号，如汉语中使用的符号除了汉字本身外，还有标点符号等等。英语中用到的符号有 26 个大、小写字母，标点符号，数字符号等等。那么 BASIC 语言中需要哪些符号呢？

2.2.1 字符集

所谓 BASIC 语言的字符集，就是 BASIC 语言中可能用到的所有符号。根据符号的性质，可以分为以下几类：

1. 字母

BASIC 语言中要用到的字母有：A，B，C，D，E，F，G，H，I，J，K，L，M，N，O，P，Q，R，S，T，U，V，W，X，Y，Z。字母必须用大写。

按下键盘上的 Caps Lock 键可以控制字母为大写。

2. 汉字

中文方式下的 BASIC 语言可用全点阵的一二级汉字库中的 6763 个汉字。

3. 数字

BASIC 中可用的数字符号有：0，1，2，3，4，5，6，7，8，9 共十个数字符号。

4. 其他符号

BASIC 语言中常用的其他符号有：＋－＊/＾ （）＝＜＞，$ ％ · ：；″？ 等等。

BASIC 语言能够使用的符号都列在键盘上，请参看图 1.1。

2.2.2 BASIC 语言的保留字

在 BASIC 语言中，为了描述问题的需要，某些字符已经结合在一起，形成一个单词如 INPUT，FOR，TO ，DATA 等等，这些单词在语言中已有固定的含义并保留作专门之用，称为保留字。顾名思义，保留字就是保留给系统专用的单词，用户不得再派用场。BASIC 语言的保留字如下：

&

ABS	AND	ASC	AT	ATN
CALL	CHR $	CLEAR	CLOLOR＝	CONT
	COS			
DATA	DEF	DEL	DIM	DRAW
END	EXP			
FLASH	FN	FOR	FRE	
GET	GOSUB	GOTO	GR	
HCOLOR＝	HGR	HGR2	HIMEM：	HLIN
	HOME	HPLOT	HTAB	
IF	IN#	INPUT	INT	
	INVERSE			
LEFT $	LEN	LET	LIST	LOAD
	LOG	LOMEM：	LG	
MID $	MUSIC			
NEW	NEXT	NORMAL	NOT	
	NOTRACE			
ON	ONERR	OR		
PDL	PEEK	PLOT	POKE	POP
	POS	PRINT	PR#	PLAY
READ	RECALL	REM	RESTORE	
	RESUME	RETURN	RIGHT $	
	RND	ROT＝	RUN	
SAVE	SCALLE＝	SCRNC	SGN	
	SHLOAD	SIN	SPC(	
	SPEED＝	SQR	STEP	STOP

	STORE	STR $		
TAB(	TAN	TEXT	THEN	TO
	TRACE			
USR				
VAL	VLIN	VTAB		
WAIT				
XPLOT	XDRAW			

2.3 常量、变量和字符串

2.3.1 常量及其写法

什么叫常量？所谓常量(或称常数)就是其值是固定不能变化的量。例如:数值 195,一经写定其值就不能再变成 352 了。根据常量的书写形式,常量可分为数值常量、字符串常量和逻辑常量。

1. 数值常量

所谓数值常量就是其值是固定不变的且组成符号仅由数字、正号(＋)、负号(－)、小数点(·)和指数符号 E 按一定规则组成。

例 2.2　＋18.73,－39.265,1988,1.285E＋3(E 代表底数 10,即为 1.285×10^3)。

这些数都是正确的数值常数。数值常数的书写除了指数必须用 E 来表示底数 10 以外,原则上同数学中一样。在书写数值常量时必须注意＋、－、E 符号的正确位置。数值常量的值就是它所表示的代数值。

2. 字符串常量

字符串常量是用引号“”括起来的字符序列。

例 2.3 “我是一个学生”,“ax+by”,“abcdef”,“this is a …”,“53+24=77”,“W(5,3)”,“1988”

以上均是正确的字符串常量。在书写字符串常量时,引号必须成对使用。如“ax+b 和 ax+b”都不是正确的字符串常量,因为它们各自都只有一个引号。字符串常量的值是指两个引号之间的字符(包括字符间可能有的空格)。如“He is a student”字符串的值是 He is a student。

3. 逻辑常量

BASIC 语言中还有一种量,它专门用于描述某种条件是否成立。这种量称为逻辑常量,它用于代表真或假。在计算机内部,通常用 1 代表真,用 0 代表假。

2.3.2 变量

在数学中我们常常用 X、Y 等字母来表示一些未知的量。这种未知量其值是可以变化的。例如 X,在某种情况下其值可以是 3.125,在另一种情况下它又可以是 193.24。类似 X、Y 的这种未知量,在 BASIC 语言中称为变量。根据变量的取值情况,变量又可以分为数值型变量、字符串变量和逻辑型变量。数值型变量只能取数值为其值,字符串变量只能取字符串为其值,逻辑型变量只能取真或假值为其值。

在 BASIC 语言中,如何表示常量和变量呢?表示常量很直观,只需把常量写在需要的地方就行了。而要表示变量就必须先给变量取个名字,我们称为变量名。变量名是用字母开头的字母或数字的序列。例如前面的 A,B,H 就是三个不同的变量名。

例 2.4 X,Y,X1,Y5,AB,H,S,T1

这些都是正确的变量名。而 A—5,X)3.5,T+5 都不是正确的变量名。在给变量取名时必须注意以下几点:

①变量名必须用字母开头,其后可以紧跟一个字母或一个数字符号。如果一个变量名超过了两个字符,那么只有前面两个字符有意义,如 ABD 和 ABCDE,均把 AB 作为变量名,因而 ABD 和 ABCDE 实际上是同一个变量名。

②变量名的组成符号除了英文字母和数字符号以外,不能夹杂其它字符。初学者容易忽视的是常常在变量名中夹杂空格,这是不正确的。也可以直接用汉字作变量名。

③BASIC 保留字不能再用作变量名。

只要遵守以上规则,对变量的取名就是自由的了。建议在给变量取名时最好有一定的含义。如时间用 T,高度用 H,面积用 S 等等,使人们一目了然。

2.3.3 数据类型

根据变量取值的范围,变量又可以分为三种:实型变量、整型变量和字符型变量(或称串变量)。实型变量取值为实数,即带有小数或指数形式的数值。整型变量取值只能是整数,即仅由数字、+、—符号组成的数。串型变量只能取值为字符串。

1. 数据类型的标志

对于一个变量名,为了明确表示其数据类型,在构造变量名时可以给出一定的标志。BASIC 语言约定如下:如果一个变量名之后紧跟一个%符号,则表示该变量是整型变量。如 A%,X%等。如果变量名之后紧跟的是 $ 标志,则意味着该变量是串型变量。一个实型变量就是变量名本身,即变量名之后不跟任何标志。

一个变量究竟应该属于什么类型,实际使用中不难划分。总

的原则是:如果一个变量要参与算术的加、减、乘、除以及乘方运算,那么肯定不能是串型变量,多数情况下还是取实型变量为好,若取整型变量,运算结果的小数部分有可能被丢掉。

2. 数值的表示范围

对于一台具体的机器来说,表示数值的大小总是有限的。但不同类型的机器能够表示的数值范围的大小亦不相同。对于中华学习机 CEC-Ⅰ来说,其表示数值的范围如下:

|实数|$\leqslant 10^{38}$,如果实际数值的绝对值大于 10^{38},则会装不下,这种情况称为上溢。如果数值的绝对值小于 10^{-39},则当成溢出处理,这种情况称为下溢。

|整数|≤32767,整数取值只能在-32767 和+32767 之间。

串型变量取值至多 256 个字符。当然可以一个字符也没有,这种串型变量称为空串。

为清楚起见,请看表 2.1。

表 2.1 数据的类型标志及其取值范围

数据类型	标 志	例 子	取值范围
实型量	无	A	$\|A\| \leqslant 10^{38}$
整型量	%	B%	|B%|≤32767
串型量	$	C$	0~255 个字符

2.3.4 数据的输出格式

在 CEC-BASIC 中,系统根据数据的取值情况,自动确定数值数据的输出格式,其原则是:

1. 如果输出的是一个负数,则在数据前面带一个负号“—”。如果是正数,则输出数据不带任何符号(即是一个无符号数,它表示一个正数)。

2. 若 0≤|数值|≤999999999 且为整数，则作为整数原样输出。

3. 若 0.01≤|数值|≤999999999.2 且为实数，则按实数原样输出。

4. 若数值不属于 2,3 种情况，则按指数形式（又称科学计数法）输出。输出格式为：

SX.XXXXXXXXEVTT

其中：S 是数的符号（正号或负号），可以是＋或－。X 为 0～9 之间的数字，· 为小数点，E 为底数 10，V 是指数的符号（＋或－），TT 为指数值。

例 2.5 若 A＝0.000009，输出格式为 9E－6

若 B＝1234567899999999，输出为 1.23456789E＋15

需要特别注意的是：

①BASIC 输出数据时至多只能达到 9 位有效数字，输入数据时可以超过 9 位数字。

②BASIC 对计算结果的舍入原则是：当不足 9 位时给出精确值，达到或超过 9 位时给出 9 位数的近似值，在第 10 位上进行四舍五入。

③当一个数被转换成整型数时，则取小于或等于该数的最大整数作为转换后的整数结果。因此，0.999 转换成整型数是 0，而－0.05 转换成整数则是－1。

2.4 函数

为使实际应用方便起见，BASIC 把诸如数学中的三角函数，自然对数，开平方，求绝对值等运算用专门的符号表示出来，这就叫函数。当用户需要时就在程序中直接调用。根据函数的用

途可分为数学函数、字符串处理函数、类型转换函数等等，详细情况请参见第七章有关节段。函数的一般形式如下：

〈函数名〉(〈形式参数〉)，其中函数名是 BASIC 语言给函数取的名字，形式参数是调用时必须提供的参数。由于这里仅起形式上的作用，因而称为形参，有些资料又称哑元。调 用时实际提供的参数才起真正的作用，因而称为实在参数(简称实参)，又称实元。下面列出常用的几个函数：

SIN(X)—求 SINX 的值。

COS(X)—求 COSX 的值。

ABS(X)—求 X 的绝对值。

SQR(X)—求正数 X 的平方根($\sqrt{X}$)。

EXP(X)—求 e^x 的值。

LOG(X)—求 X 的自然对数值。

INT(X)—取 X 的整数部分值。

实际应用中要注意：

①凡是能够用函数来表示的运算最好用函数运算。

②所有三角函数的参数都用弧度表示。如果提供的是角度，那么必须转换成弧度，转换公式是：$\pi=180°$。

③X 可以是表达式(参见下一节)。

④圆括号不能少掉。

⑤函数运算后一定能够得到一个值。

⑥由 BASIC 语言提供的函数称为标准函数。

2.5 表达式

我们已经讨论了常量、变量和函数。如果把这些量用相应的运算符按一定的规则联系起来就构成 BASIC 语言中所谓的表

达式。表达式可以分为四种:算术表达式,关系表达式,逻辑表达式和字符串表达式。

2.5.1 运算符及其优先级别

1. 算术运算符

BASIC 中要用到的算术运算符有:

+, —, *, /, ^

加, 减, 乘, 除, 乘方

算术运算符可以用来构成算术表达式以进行算术运算,其运算规则与数学中一样。首先进行乘方,其次乘除,最后进行加减运算。如果有圆括号,那么括号优先。同级运算从左到右依次进行。算术运算的结果必定是一个数值。

2. 关系运算符

<, <=或=<,=, >, >=或=>, <>或><

小于,小于或等于,等于,大于,大于或等于, 不等于

关系运算符用以对两个算术项进行比较,结果产生一个逻辑值真或者假。CEC-BASIC 用 1 代表真,用 0 代表假。由于 BASIC 中不允许把几个关系运算符连在一起,因而不存在优先级关系,如 a>b>c=d 的写法是不正确的,它必须换成逻辑表达式来书写。

3. 逻辑运算符

BASIC 的逻辑运算符有:

NOT	AND	OR
非	与	或

逻辑运算符用于进行逻辑运算,运算结果仍然是一个逻辑值真或假,运算规则如下表所示。假设 A、B 是取值为真(1)或假(0)的逻辑变量,A 和 B 的逻辑运算结果也示于表 2.2 中。其中

1 代表真值,0 代表假值。

表 2.2 逻辑运算

A	B	NOT A	A AND B	A OR B
1	1	0	1	1
1	0	0	0	1
0	1	1	0	1
0	0	1	0	0

从表 2.2 的运算结果可以看出逻辑运算符的运算规则有如下的规律:

NOT A 是对 A 的否定。如果 A 为真,那么 NOT A 的结果是假,如果 A 为假,NOT A 的结果为真。

对于 AND 运算,只有当 AND 的两端同时为真(即 A 为真 B 为真)时,A AND B 的结果才为真,其余情况均为假。

对于 OR 运算,只有当 OR 两端的值同时为假时,A OR B 的值才为假,其余情况均为真。

逻辑运算符的优先级别是:NOT AND OR。

2.5.2 算术表达式

1. 什么叫算术表达式?

算术表达式是指一个数值常量、数值型变量或算术函数以及用算术运算符＋－ * / ^ 和圆括号()将这些量按一定规则连接而成的式子,我们都可以称之为算术表达式。一个数值常量或数值变量或算术函数是算术表达式的特殊情况。为了区分,有时将它们称为简单算术表达式,而把带有运算符的表达式称为复杂算术表达式。在不致混淆的情况下一律称为算术表达式。

例 2.6 1988,－3.72,1.534E＋10,X,Y1,A%,SIN(3)

这些都是简单算术表达式的例子。

例 2.7　B * B－4 * A * C，(1＋SIN(X) ^ 2/(Y * Y＋SQR(1＋2 * Y＋5 * Y ^ 5))，

这是两个复杂算术表达式。从理论上来说，利用算术运算符和圆括号，我们可以把一个算术表达式构造得非常复杂。

2. 算术表达式的书写规则

利用 BASIC 语言来进行数值计算，必须将数学式子写成 BASIC 所允许的算术表达式的形式。其书写原则是：

①把乘号"×"写成"*"，并且不能省写或简写。除号"÷"写成"/"。

例 2.8　将下列数学式写成 BASIC 算术表达式。

$$\frac{(A+B)\sqrt{X^2+1}}{C+D}$$

⇒(A＋B) * SQR(X * X＋1)/(C＋D)

②注意成对使用圆括号，不论需要用到多少重括号，算术表达式中只能使用圆括号()。

例 2.9　将下列数学式写成算术表达式。

$\left(\frac{A+B}{X+Y}\right)^3$ ⇒((A＋B)/(X＋Y)) ^ 3

③数学式中用到的三角函数、自然对数、e^x 指数函数、开平方根等运算，都必须换成 BASIC 的相应函数表达式。

例 2.10　将下列数学式写成算术表达式。

$$\frac{\mathrm{Sin}e^{ax}+\ln bx}{-a\sqrt{X+A+1}}$$

⇒ABS((SIN(EXP(A * X))＋LOG(B * X))/(－A * SQR(X＋A)＋1))

④区分容易混淆的字符。如字母 I 和数字 1，字母 O 和数字

0 等。

3. 算术表达式的运算规则

算术表达式一经确定，就可以按给定的规则进行运算以得出一个具体的数值。BASIC 的算术表达式运算规则同数学中一样：括号优先，有多重括号时由内到外。乘方、乘除、最后是加减运算。同级运算按从左到右原则处理。

4. 算术表达式的值

算术表达式的值就是将变量值代入算术表达式后按给定运算规则运算而得到的代数值。这个值必定是一个数值，它的数据类型一般来说是实型。

2.5.3 关系表达式

关系表达式是用关系运算符将两个算术表达式或字符串表达式连接而成的式子。其一般形式如下：

〈算术表达式〉〈关系运算符〉〈算术表达式〉或〈串表达式〉＝〈串表达式〉

例 2.11　B∗B－4∗A∗C＞0，X＞Y，H＜＝1.72，“NO”＝“YES”

这些都是正确的关系表达式的例子。而 A＞B＞C 却不是正确的关系表达式。它实质上应该写成逻辑表达式的形式。

在关系表达式中，首先进行算术运算，得到算术表达式的结果后再进行比较，最后得出一个逻辑值。在例 2.11 中，假设 A＝1，B＝2，C＝3，那么将 A，B，C 的值代入关系表达式 B∗B－4∗A∗C＞0 中得：2∗2－4∗1∗3＞0，显然其值为假(0)。这就是说，在特定条件(A＝1，B＝2，C＝3)下，B∗B－4∗A∗C＞0 是不成立的。关系表达式通常用于构造某种选择条件。

在所有关系运算符中，对等号“＝”的使用要特别小心。有时

我们认为不等的数（如 10^{-39} 和 0），由于机内转换或其它处理等原因而造成两个数的相等，导致条件的满足或不满足，致使发生我们难以料及的错误。

对于用“＝”连接的两个字符串，只要等号两端的字符相同，则其结果为真，否则为假。如：“ABC”＝“ABC”的值为真，而“ABC”＝“XYZ”的值为假。

2.5.4 逻辑表达式

一个关系表达式或用逻辑运算符 NOT、AND、OR 将关系表达式按一定规则连接而成的式子都称为逻辑表达式。

例 2.12　A＞B AND C<D，NOT X＝Y AND B＊B－4＊A＊C＞0。

这些都是正确的逻辑表达式。关系表达式仅是逻辑表达式的特殊情况。对于数学中的 A＞B＞C 的写法，写成逻辑表达式应该是 A＞B AND B＞C。

在一个比较复杂的逻辑表达式中，可能同时包含多种运算符，其运算优先级是这样规定的：首先进行算术运算，其次进行关系运算，最后进行逻辑运算。在各自的运算中又按各自规定的优先级进行运算。逻辑表达式最终一定能够得出一个逻辑值。

例 2.13　求下列逻辑表达式的值（设 A＝1，B＝2，C＝3）。

B＊B－4＊A＊C＞0 AND(NOT 5＞A) OR (A＋B)＊(C－B)＜ ＞B^3 AND A＋B＞A＊B

⇒2＊2－4＊1＊3＞0 AND(NOT 5＞1)OR(1＋2)＊(3－2)＜ ＞2^3 AND 1＋2＞1＊2

⇒-8＞0 AND(NOT 5＞1)OR 3＜ ＞8 AND 3＞2

⇒0 AND NOT 1 OR 1 AND 1

⇒0 AND 0 OR 1 AND 1

$\Rightarrow 1$

按照给定的运算规则运算后其结果为逻辑值真(1)。

需要强调指出的是:由于系统用数字 1 代表真,用 0 来表示假,因而在算术表达式中夹杂逻辑表达式是可以的,计算机能够按规则计算出正确的结果。如:

假设:A=1,B=2,C=3,则

$A*(A<C)*5+10*(C>B)-(B*B-4*A*C>0)$

$\Rightarrow 1*(1<3)*5+10*(3>2)-(2*2-4*1*3>0)$

$\Rightarrow 1*1*5+10*1-0$

$\Rightarrow 15$

注意:如果逻辑表达式不带圆括号,那么首先进行算术运算,得到的结果再进行关系运算或逻辑运算。上式中由于有圆括号,因而应当首先计算,得到结果值 1 或 0 再与算术表达式进行加、减、乘、除和乘方运算。

2.5.5 字符串表达式

字符串表达式比较简单,它只需用串运算符"+"将两个字符串(或串变量)或串函数连接起来,从而生成一个新的字符串。

例 2.14 将字符串"ABCD"和"XYZ"连接起来。

"ABCD"+"XYZ"生成一个新串"ABCDXYZ"。

2.6 什么叫 BASIC 程序

前面比较详细地介绍了 BASIC 语言中经常涉及到的常量、变量、字符串、函数和表达式等基本概念。下面再来说明什么是 BASIC 程序。所谓 BASIC 程序,就是用 BASIC 语句书写的完成某种计算或事务管理的详细步骤,请看下面的例子。

例 2.15　求一元二次方程 $ax^2+bx+c=0$ 的根。

由数学知识我们知道，求解一元二次方程根的公式为：

$x_{1,2}=\dfrac{-b\pm\sqrt{b^2-4ac}}{2a}$，判别式 b^2-4ac 如下：

$$
若\ b^2-4ac\begin{cases} >0, & 则有两个不同实根。\\ =0, & 则有两个相同实根。\\ <0, & 则有共轭虚根。\end{cases}
$$

根据一元二次方程的求解方法，编成如下的 BASIC 程序：

```
10  INPUT A,B,C
20 D = B * B - 4 * A * C
30  IF D > 0 THEN 80
40  IF D = 0 THEN 110
50 X1 =  - B / (2 * A)
60 X2 =  SQR ( - D) / (2 * A)
65  PRINT "共轭虚根"
70  GOTO 130
80 X1 = ( - B +  SQR (D)) / (2 * A)
90 X2 = ( - B -  SQR (D)) / (2 * A)
95  PRINT "两个不等实根"
100  GOTO 130
110 X1 =  - B / (2 * A)
120 X2 = X1
125  PRINT "两个相等实根"
130  PRINT "X1=";X1;"  ";"X2=";X2;
140  END
```

```
]RUN
?5,6,4
共轭虚根
X1=-.6   X2=.663324958
```

从上例可以看出，一个 BASIC 程序包括了这样几个部分：

1. 语句行号

这是每个语句的开始符号。它由至多 5 位（0～63999 之间）的无符号整数组成。语句行号数值的小和大，就代表了语句在程序中的前与后，也表示语句执行的先与后，一般来说，BASIC 程序从行号最小的语句开始依次执行。前后两个行号并不要求连续。

2. 语句

语句是 BASIC 用以完成操作动作的一些有专门意义的字符序列。如 INPUT 表示输入，PRINT 表示打印输出。语句必须按照 BASIC 语言的语法规则来书写，而书写语句的顺序要按完成具体任务的要求依次（体现在语句行号的小和大上面）书写。在书写语句时，可以一个语句占用一个书写行，也可以几个语句写在同一行上。如 110 和 120 两句，可以写在一行上：

110 X1=-B/(2*A)：X2=X1↙

3. 回车键

如果要把程序输入到机内，那么每个语句行输入完后必须敲一个回车键 Return 。为写起来方便，我们用↙表示回车键。回车键是一个语句行完结的标志。在书写程序时可以不写↙，但输入程序时必须敲↙键。

4. 程序结束标志 END

在一个程序中，程序的结束标志 END 可写也可不写，这不

影响程序的正确性(但用子程序时有例外)。严格说来,END 不是 BASIC 的语句,它是操作系统提供的实用命令。

5. 源程序

源程序就是用 BASIC 语言书写的程序,亦称 BASIC 源程序,或称源程序,通常简称为程序。从以上的例子我们可以得出如下的结论:

一个程序由若干个语句组成;一个语句由若干单词组成(这些单词包括保留字、变量、常数、函数或表达式);一个单词又由若干字符组成。这同自然语言一样,字符组成字,字组成句,句组成文。一个程序就相当于是用 BASIC 语言写的一篇文章。我们研究 BASIC 语言,实际就是讨论由字组成句,由句组成文的规则。

在编写和输入 BASIC 程序时必须注意:

①一个语句总是以行号开始,以↙作为结束。

②语句行号之间最好留有一定的间隔数,以利于插入新行。如 10,20,30 行号比 10,11,12,13 行号取得好。

③如果一个语句行号上需要写多个语句,那么语句间必须用冒号":"隔开。

④一个语句行连同字符和空格不能超过 239 个字符。如果一个语句可能超过 239 个字符,那么可以拆成多个语句。

⑤用于构成变量名、保留字和命令中的字符通常用大写。

⑥BASIC 程序的执行总是从最小的行号开始依次执行(除非有转向)。因而程序中的行号不能随便乱写。

2.7 BASIC 程序的输入

输入 BASIC 程序之前必须事先进入 BASIC 系统。

2.7.1 如何进入 BASIC 系统

这里就输入程序前如何进入系统给出简要说明。

1. 固化 BASIC 系统的启动

请先按第一章介绍的方法连接好中华学习机的硬件系统。打开显示器电源,再打开主机电源开关。屏幕显示:

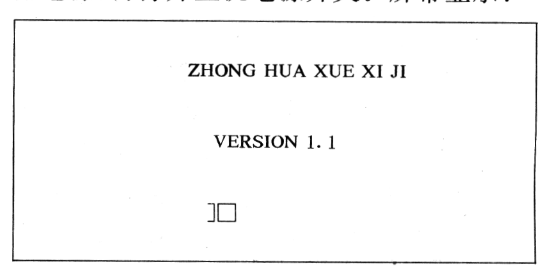


图 2.1 系统进入 BASIC 状态

这时主机自动进入西文 BASIC 状态。“]”是 BASIC 系统的提示符,光标紧跟其后闪动,表示机器已作好准备接收输入字符。

2. 用磁盘操作系统来启动 BASIC 系统

前面介绍的是直接在主机上启动固化 BASIC 语言。如果我们要用磁盘上的操作系统 DOS 3.3 来启动,那么可按如下步骤操作:

①打开磁盘驱动器中央的盖子,把 DOS 3.3 的软盘带有标签的一面朝上,试探性地轻轻插入到驱动器中,然后关好驱动器的小门。

②打开显示器和主机电源,稍候,屏幕显示:

```
DOS VERSION 3.3 08/25/80
APPLE Ⅱ PLUS OR ROMCARD SYSTEM MASTER

(LOADING INTEGER INTO LANGUAGE CARD)

]□
```

图 2.2　用 DOS 3.3 进入 BASIC 状态

此时系统进入 BASIC 状态。“]”表示可以输入字符了。以上启动方式称为冷启动。

③如果开机之前没有把 DOS 3.3 软盘插入到驱动器中，那么系统直接从主机进入固化 BASIC 状态。如果这时需要装入 DOS 3.3 操作系统，那么把 DOS 3.3 软盘插入驱动器后，在“]”提示符后敲入：

]PR＃6↙，这时仍然可以进入 BASIC 状态，屏幕显示仍如图 2.2 所示。（这称为热启动）

3. 系统复位

如果主机电源已经打开，可以用敲 Ctrl - Reset 键（先按住 Ctrl 键再敲 Reset 键）的办法启动系统。这通常也称为热启动，亦叫系统复位。

2.7.2　中西文方式的转换

系统启动以后便自动进入西文使用方式。如果我们希望进入中文方式，那么最简单的方法就是敲键盘上的 中文 键，屏幕显示如图 2.3 所示。

这时系统便处于中文使用方式下的字母输入状态。如果要在中文方式下输入非汉字的字符，那么就必须处于“字母”状态

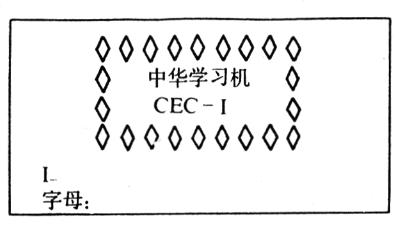


图 2.3

下。如果需要输入汉字，就必须转到相应的汉字输入方式。下面三个键经常用到：

F1——允许在中文方式下输入字母和字符。

F2——允许用汉语拼音方式输入汉字。

F3——允许用区位码方式输入汉字。

事实上，进入中文系统有三种方式：

1. 直接敲键盘上的[中文]键。这适合在对话方式下使用。

2. 在 BASIC 或 DOS 状态下敲入 PR＃3。如果是在程序中希望进入中文系统，那么必须用语句的形式写成：〈行号〉 PR＃3：PTINT。

3. 在监控程序下敲 3 CTRL＋P 或 3 CTRL＋K 键也可以进入中文系统。退出中文系统也有三种方式：

1. 直接敲键盘上的[西文]键。

2. 在 BASIC 程序中用〈行号〉 PRINT CHR $ (17)：PRINT。

3. 在监控方式下敲 C33AG 命令。

不论是进入还是退出中文系统，内存的程序均不受影响。

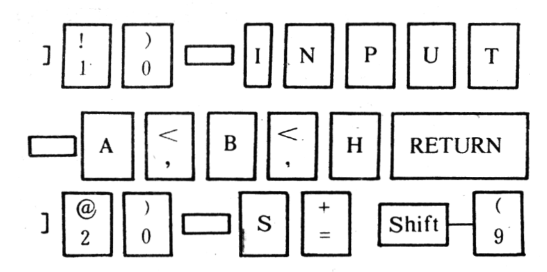
2.7.3 如何输入程序

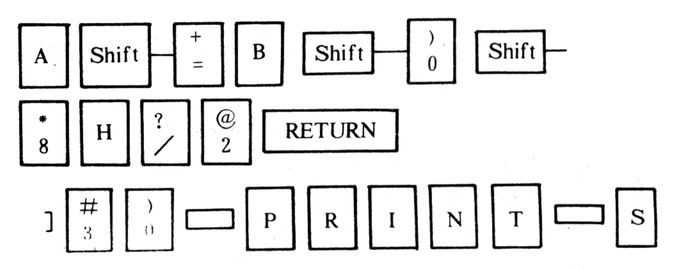
1. 西文字符的输入

如果我们仔细观察键盘就会发现，有的键上只标有一个字符，如26个字母键就是如此。而有的键上却标有两个字符，如10个数字键，除了标有数字以外，还标有其它字符。如 *8 键，上部标的是 *，下部标的却是8。那么敲该键时如何区分呢？系统是这样规定的：如果待输入的符号所在的键上只有一个符号或是键的下部符号，那么就直接敲键。如要输入AB32，就直接依次敲 A B #3 @2 四个键，屏幕显示的就是AB32。这种情况称为见码敲键。如果待输入的字符是键的上部字符，那么在敲该字符键时必须先按住换档键 Shift，然后再敲上部字符所在键。如要输入字符 *，就必须先按住 Shift 键，然后再敲 *8 键。这种情况称为换档敲键。记成 Shift - *8。凡是需要输入上部字符时都必须用换档敲键法。输入单个字符或下部字符时一律用见码敲键法。为便于读者练习，我们以下列程序的输入为例：

```
10  INPUT A, B, H ↙
20  S=(A+B) * H/2↙
30  PRINT S↙
```

敲键动作如下（ 表示空格键）：





其中每行开头出现的“]”是系统提示符，我们不能敲入。

2. 汉字的输入

中华学习机上目前提供了三种输入汉字的方法。一是汉语拼音，二是区位码，三是五笔字型输入法（见第二册《汉字处理与数据库技术》），这里简介拼音输入法。

我们以输入下列程序为例：

10 PRINT“中华学习机”↙

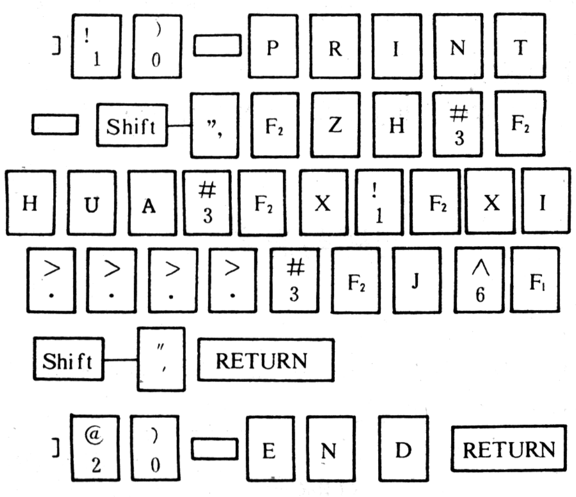
20 END↙

首先，敲[中文]键进入中文方式后系统自动处于“字母”状态，我们可以输入行号10和PRINT″。下面要输入汉字，因而敲[F_2]键进入拼音状态，敲入“中”的拼音ZH，屏幕显示：

拼音:zh 1 这　2 主　3 中　4 种　5 着　6 争

一看便知道需要的“中”字出现了。这时只要敲汉字前面的序号3，就把“中”字取到内存并显示在屏幕的相应位置上。凡是需要用拼音输入汉字都必须敲[F_2]，使其转到拼音输入法的开始状态。而凡是输入其它符号都必须敲[F_1]键使其回到字符状态。上面的叙述过程可以用下面的敲键动作来表示：

][中文]



汉字的查找输入过程如下：

状态键	状态	拼音	供选择的汉字	敲键取字
F2	拼音：	zh	1这 2主 3中 4种 5着 6争	#/3 中
F2	拼音：	hua	1花 2哗 3华 4猾 5滑 6画	#/3 华
F2	拼音：	x	1学 2下 3小 4线 5现 6性	!/1 学
F2	拼音：	xi	1袭 2席 3习 4媳 5喜 6铣	#/3 习
F2	拼音：	j	1就 2级 3阶 4进 5加 6机	^/6 机

敲 [〉/.] 键表示在同音字中继续查找后继汉字，敲 [〈/,] 键就倒回到先前显示的汉字。

本节所述内容经常用到，建议读者面对一台中华学习机，亲自上机练习。

问题与思考

1. 为什么变量名必须用字母开头？
2. 为什么算术表达式中的乘号既不能省略也不能简写？
3. 为什么要注意成对使用圆括号？

习题二

1. 分别写出三个不同的数值常数和字符串常数。

2. 下列字符序列哪些是正确的 BASIC 变量名？

XYZ，S1，END，S－1，SQRT，SUM，DATA，DTE

3. 写出下列数据的输出格式。

－328.75，＋4235，$73.48_{10}{}^{5}$，－1234567892222

4. 将下列数学式写成 BASIC 表达式。

①$\frac{n\ (n-1)^2}{2}$

②$\sqrt{X^{a+b^c}-\frac{a}{b\cdot c}}$

③$\sqrt[3]{Ln\ \sqrt{1+d^2}-e^{2\beta}}$

④$2RSin\ \frac{\beta}{2}$

⑤ $\dfrac{\cos x+\left|\dfrac{a-b}{a+b}\right|\cdot X^{5}}{e^{a^{b}}+e^{a}b^{x}}$

⑥ $\dfrac{\dfrac{a-b}{a+b}}{a-\dfrac{a+1}{a-\dfrac{a+1}{b}}}$

5. 在机器上练习下列程序的输入。

```
10 INPUT A, B, C$↙
20 D=A+B↙
30 C$ ="我正在使用计算机"↙
40 PRINT C$ ,D ↙
```

请写出输入程序时的敲键动作。

第三章　简单的 BASIC 程序

前已述及,BASIC 程序是由若干语句组成的。那么 BASIC 语言中有些什么语句呢?它们的书写格式、作用如何呢?如何用语句来编写程序呢?从本章起,我们将由浅入深地介绍 BASIC 的一些基本语句以及如何用这些语句来编写完整的能够在 CEC-I 机上运行的程序。

3.1　赋值语句

在 BASIC 程序中,用得最多的语句可能就是赋值语句了。

3.1.1　赋值语句的格式和作用

格式:[LET]〈变量名〉=〈表达式〉

作用:将表达式的值送到变量中保存起来。

其中:LET 是赋值语句的标志,实际使用中可以省略不写。变量名又称左部变量,它用于保存表达式的值。变量的类型可以是整型、实型或串型。“=”表示一种传送动作。表达式可以是任何合法的算术表达式或关系表达式或逻辑表达式,甚至也可以是字符串表达式。当然,“=”两端的数据类型应当一致。

例 3.1　求$\frac{3\sqrt{100}}{\mathrm{Ln}3+e^{2}}$的值。

为便于读者练习,我们写一个完整的程序(行号之前的“]”是系统提示符,由机器自动显示,程序中不写):

```
]10 W=3*SQR(100)/(LOG(3)+EXP(2))↙
]20 PRINT "W=";W↙
]RUN↙          (以下是运行结果)
W=3.5345396
```

这个程序包含了两个语句,语句行号分别为10和20。我们可以用LIST命令列出刚输入的程序。其中第10句就是一个赋值语句,它表示把3*SQR(100)/(LOG(3)+EXP(2))算术表达式的值计算出来再送到实型变量W中保存起来,以备后用。第20句是打印语句,表示把W的值显示在屏幕或打印在宽行纸上(PRINT语句后面还要专门介绍)。

RUN表示运行这段程序。运行程序之前,必须按要求将程序输入计算机内。对于这种简短的程序,建议读者直接在CEC—I机上输入,而不必用笔写在纸上后再输入。应当养成直接在计算机终端(键盘)上编写程序的习惯。

W=3.5345396是该程序运行后输出的结果。这里虽然只有两个语句,却是一个完整的能够在机器上运行的程序。

上例程序中赋值语句也可以写成:

```
10 LET W=3*SQR(100)/(LOG(3)+EXP(2))
```

例3.2

```
10 A=5↙
20 B=6↙
30 C=7↙
40 D=B*B-4*A*C↙
50 PRINT D↙
RUN↙          (以下是运行结果)
-104
```

前三句将5,6,7分别送到A,B,C变量中,第40句利用前面获得的A,B,C的值来计算B^2-4AC的值(-104)并将其(-104)送到变量D中。第50句打印出D的值。

例 3.1 和例 3.2 中赋值语句的左部变量和右端表达式都是数值型量。下面我们来看如何保存字符串。

例 3.3　将字符串连接后打印出来。

```
10 H1$ ="张华"↙
20 H2$ ="是"↙
30 H3$ ="一个好学生。"↙
40 H$ =H1$ +H2$ +H3$ ↙
50 PRINT   H$ ↙
RUN↙              (以下是运行结果)
张华是一个好学生。
```

10、20、30 三个赋值语句的作用是把相应的汉字送到各自的字符串变量中。第 40 句是将三个字符串变量(H1$、H2$和H3$)的值进行连接运算,运算的结果仍然是一个字符串"张华是一个好学生",再将该字符串送到变量 H$ 中。第 50 句打印出串变量 H$ 的值(张华是一个好学生)。从这个例子也可以看出字符串运算符"+"和算术运算符"+"的本质区别。

从前面几个例子来看,行号均是从 10 开始的。提醒读者不要造成这样一个错觉:似乎 BASIC 程序的行号只能从 10 起编号。只要行号在 63999 以内,用任何无符号整数开始作行号都可以。

语句行号的另一个规律读者可能早已发现。这就是行号始终是由小到大编号。这是正确的,也是必须的。BASIC 语言中先执行的语句必须具有较小的行号。

对于程序的输入,我们可以按照语句行号由小到大的顺序依次输入机内,也可以随心所欲地按自认为合适的顺序输入机内。只要输入正确,对计算机来说运行效果完全一样。对于例 3.3 的程序,我们也可以按如下的语句顺序输入。

例 3.4 不按行号小大输入的程序。

```
40 H$=H1$+H2$+H3$
10 H1$="张华"↙
50 PRINT H$↙
30 H3$="一个好学生。"↙
20 H2$="是"↙
RUN↙          (以下是运行结果)
张华是一个好学生。
```

运行结果完全一样。为什么两个相同程序排列顺序完全不同以后运行的结果仍然相同呢?有两个原因:其一是语句行号没有变,只是输入语句的先后次序变了;其二是计算机总是从行号小的语句按行号由小到大顺序依次执行,尽管输入次序打乱了,但计算机执行语句的顺序仍然没有变。所以运行结果完全相同。如果我们把行号和语句的对应关系打乱了,那运行结果一般说来是不同的。

例 3.4 的作法尽管是正确的,却实在是不可取的,何必自己和自己过不去呢!

3.1.2 赋值语句的执行步骤

赋值语句写起来是简单的,但计算机系统在实现该语句功能时却必须采取三个步骤才能完成。

1. 确定左部变量

程序中给出一个赋值语句后,系统首先就要确定其左部变量。以上赋值语句中给出的左部变量都是简单变量的情况,因而确定左部变量并不麻烦。如果左部变量是下标变量(第六章介绍数组时将有详细说明),那么就必须先计算出下标后才能确定左部变量。在计算机系统内部,确定下标变量这种左部变量时还有一番复杂的计算,这里从略。有兴趣的读者请参阅编译技术方面

的书籍。

2. 计算右端表达式的值

如果右端表达式是一个算术表达式,则按算术表达式的规则进行算术运算,最终得出一个数值;如果右端是字符串表达式,那么就按照字符串表达式的运算规则计算其值;右端也可以是逻辑表达式,相应计算的值就是逻辑值。下面是逻辑表达式作为右端项的例子:

例 3.5　右端项是逻辑表达式的情形。

```
10 A = 8
20 B = 10
30 C = 7
40 L = A > 4
50 T = A * A + B * B > C ^ 4 AND L OR  NOT B < 100
60  PRINT L,T

]RUN
1                   1
```

在例 3.5 的程序中,40 和 50 两句的右端都是逻辑表达式。左部变量 L 和 T 均是逻辑变量。这个例子告诉我们,也可以把逻辑表达式的值送到一个变量中保存起来,以后需用时就直接引用该逻辑变量。

3. 将表达式的值传送到左部变量中保存

计算机要采取的第三个步骤就是将第二个步骤中计算出来的值送到由第一步所确定的左部变量中保存起来,后面的语句就可以直接引用变量名来使用其值了。

以上三个步骤,完全由计算机的硬件和软件来完成,不需要

我们人工干预。在实际使用中，我们只需按要求写出正确的赋值语句就行了。事实上，机器在执行这三个步骤时，从直观上我们是感觉不到的。

3.1.3 使用赋值语句时的注意事项

赋值语句在实际程序设计中用得最多，几乎任何一个有意义的 BASIC 程序都不可能没有赋值语句。但使用时必须注意以下几点：

1. 左部变量和右端表达式的数据类型必须一致

如果右端是算术表达式，那么左部变量只能是数值型变量。如果右端是字符串表达式，那么左部变量只能是串型变量（标志是 $）。如果右端是逻辑表达式，那么左部变量只能是逻辑变量。逻辑变量同实型变量一样没有任何外观标志，但系统内部是清楚的。

如果赋值语句等号两端的数据类型（主要是指算术类、逻辑类和字符串类）不一致，那么执行该语句时就会发生数据类型不匹配的错误。

例 3.6 执行下面的赋值语句。

```
10 D$ =3+LOG(5)↙
20 PRINT D$ ↙
RUN↙                    (以下是运行结果).
? Type mismatch error in 10
```

运行这个程序后，在第 10 句发生了类型不匹配的错误(Type mismatch error in 10)。我们重新检查程序就会发现，在第 10 句的赋值语句中，左部变量是串型变量，而右端却是一个算术表达式，理所当然地要发生数据类型错误。改正的办法是根据实际情况将左部变量换成实型量或将右端写成字符串形式。

如果赋值语句的两端均是算术量（仅仅是整实之间的差别），那么机器将自动进行这种转换。转换的原则是：如果左部是整型 量（标志是%）而右部是实型量，那么机器自动将右部实型量取整后按整数形式保存到左部变量中。（当一个实型数转换成整型数时，取小于或等于该实数的最大整数作为转换后的结果）。如果左部是实型量而右端是整型量，那么机器自动将整型值转换成实型值后保存在左部变量中。

2．左部变量只能是一个不带运算符的变量名，而不能是常数或带运算符的表达式

3．赋值语句的标志 LET 可以省去不写

前面所举的赋值语句的例子中已经这样作了。LET 省去后不会对语句有影响。

4．等号“＝”的作用

赋值语句中的等号“＝”，既不同于数学中的“＝”，又不同于关系表达式中的“＝”。数学中的等号代表的是一个相等关系，左右两端可交换。而关系表达式中“＝”代表的是一种相等比较运算，运算的结果是一个逻辑值。赋值语句中的“＝”代表的是一种传送的操作动作。请看下面的例子：

例 3.7
```
10 X=3↙
20 X=X+5↙
30 L=X=5↙
40 PRINT X,L↙
RUN↙          (以下是运行结果)
8             0
```

对于第 20 句的 X＝X＋5，在数学中看来这是完全不可能的，因为不会有任何一个变量加上一个数值后仍然等于变量本身。但 BASIC 语言是允许的，它表示在原来基础上累加一个值

后再送到原单元中保存。

第30句是一个逻辑赋值语句，它表示把X=5的逻辑值送到逻辑变量L中保存。这一句中有两个等号，前一个“=”代表传送操作，后一个“=”代表关系运算。

从输出结果来看，8是X累加上5后得到的值。而0是逻辑值假，因为X的值并不等于5，所以L的值是假(0)。

5. 可以利用赋值语句进行累加和连乘。

例3.8 运行下列程序。

```
10 A = 3
20 W = 0
30 S = 1
40 W = W + A
50 W = W + A
60 S = S * A
70 S = S * A
80  PRINT A,W,S

]RUN
3                 6                 9
```

40～50就将A的值累加起来了，因为累加了两次，所以输出结果为6。60～70句把A的值连乘了两次，所以输出结果为9。

6. 赋值语句也是使变量获得数据的一种方式。

3.2 键盘输入语句

赋值语句可以使变量获得数据这是毫无疑问的。但是，如果

把赋值语句作为变量获得数据的唯一方式，那么读者将会发现，输入 100 个不同的数据就得有 100 个不同的赋值语句，那乏味的程序不就太长了吗？是的，对实用来说可能是不方便的。因而 BASIC 语言提供了多种输入语句。

3.2.1 键盘输入语句的格式及其作用

格式：INPUT [〈提示〉;]〈变量名表〉

作用：允许用户从键盘上接收数据并送到对应变量中。

其中：待选项[〈提示〉;]是用户自己给出的提醒输入数据的标记，它由字符串组成。变量表是希望从键盘上接收数据的一个或多个变量名。如果表中有多个变量名，那么彼此之间要用逗号“,”隔开。

例 3.9 用键盘语句输入 A,B,C 的值。

```
10 INPUT "请输入数据:";A,B,C↙
20 D=B*B-4*A*C↙
30 PRINT A,B,C↙
40 PRINT D↙
RUN ↙
请输入数据:5,6,7↙
5          6          7
-104                    (运行结果)
```

对于这个程序需要作如下的说明：

①“请输入数据”：这是为提醒用户注意而给出的提示信息。在实际应用中，用户可以给出自己所需要的提示信息。

②提示信息之后一定要有一个分号“;”，其作用是将提示信息和变量表隔开。

③INPUT 之后的 A,B,C 是变量表，表中有 A、B、C 三个变量名。A、B 和 B、C 间都用逗号隔开。

④敲入 RUN↙命令后，屏幕上显示："请输入数据："这几个字符，并等待用户输入数据。本例中输入的是 5,6,7 三个常数。至此，机器自动把 5 送到 A 中，6 送到 B 中，7 送到 C 中。输入的数据间要用逗号","隔开(请读者考虑为什么要用","隔开)。最后的 5,6,7 和 −104 是输出的运行结果(30,40 两句的作用)。

在 INPUT 语句中，如果省略了提示信息，那么系统用一个问号"?"作为提示用户输入数据的标志。参看下例。

例 3.10　运行下列程序并输入数据。

```
10  INPUT BA,A,B,H
20 S = 3 * (A * A + B * B + H * H)
30 V = BA * H *  SQR (S) / 6
40  PRINT BA,A,B
50  PRINT H,V


]RUN
?3.14159,5.3,2.7,3.72
3.14159            5.3                2.7
3.72               23.6682184
```

其中的"?"就是系统给出的提醒输入数据的信息。

3.2.2　使用 INPUT 时的注意事项

1. 变量名必须和输入的数据一一对应。

如果变量名和所敲入的数据不对应，那么系统是不容易发现这类错误的。因此，我们使用时应倍加注意，否则将导致错误的计算结果。如果有多个变量名，变量名之间必须用逗号隔开。如前面示例中的 INPUT A,B,C 所示。

2. 变量名和输入数据的类型必须一致。

如果变量名和输入数据类型不正确，则屏幕显示“？REENTER”提示，意思是要求用户重新敲入正确的数据。

3. 用 INPUT 接收字符串时，要求从键盘上敲入的字符串中不能含有逗号。因为逗号是用于分隔数据的标志。

4. 用 INPUT 接收的只能是具体的常数(可以是数值常数或字符串常数)，而不能是变量名或表达式。参看下例。

例 3.11　键盘敲入数据。

```
10 W = 5
20  INPUT "输入数据:";A,B,C
30 D = (A + B) / C
40  PRINT D

]RUN
输入数据:W,SQR(100),5+3
?REENTER
 输入数据:

Break in 20
```

显然敲入的数据不正确，因为 W，SQR(100)，5＋3 均不是常数而是表达式。

5. 如果 INPUT 中接收的是多个数据，那么在敲入数据时，可以用逗号来分隔数据，也可以用↙来分隔数据。如果用“，”分隔，最后数据之后必须敲↙键，在此情况下，↙作为所有数据是否完结的标志。

6. 如果 INPUT 中给出了提示信息，那么提示信息之后一定要有一个分号“；”，它用于把提示信息和变量表分隔开。

7. 系统约定一个算术变量在没有获得值之前其值为 0，一

个字符串变量在未得到值之前为空。

3.2.3 GET 语句

格式：GET〈变量名〉

作用：允许用户从键盘上接收一个字符到指定的变量中。这里的字符根据变量名的类型来确定，可以是数字，也可以是其它字符。

例 3.12 用 GET 将数字 3 送到 A 中。

```
10 GET A↙
20 PRINT A↙
RUN↙
3            (敲入数据字符 3)
3            (打印的结果)
```

使用 GET 来接收数据时注意：

1. 机器执行 GET 语句时屏幕上没有任何提示标记，输入的字符也不显示在屏幕上。因而可用 GET 来生成一个简单的保密码。

2. 敲入数据之后不必敲↙，因为只接收一个字符，不用敲↙标志机器也能判断数据是否结束。

3. 如果 GET 之后的变量名是算术变量，那么敲入＋、－、.、、、、:符号后变量的值均为零。而输入其它字符时均发生语法错，这时屏幕显示"？Syntax error"且程序停止执行。

4. 用 GET 来接收数据时，无论在键盘上敲入了多少个字符，变量均只能接收第一个字符。

通常，实际使用中都不大喜欢用 GET 来接收参与计算的数据(因为参与计算的数据往往不止一位数字)，而大多用于接收一个固定的字符，以便用于控制程序的执行流程。

3.3 输出语句

前面的例子中我们已经用到了输出语句 PRINT,但没有作任何解释,现在是该详细介绍的时候了。

3.3.1 输出语句的格式和作用

格式:PRINT〈表达式表〉或?〈表达式表〉

作用:将表达式的值显示在屏幕或打印在宽行纸上。程序中也可以用“?”代替 PRINT,效果完全相同。

这里的表达式表是需要打印其值的一个或多个表达式。表达式表中的各个表达式之间用逗号“,”隔开,也可以用分号“;”隔开。但用“,”和“;”隔开时,打印出来的数据的排列方式略有不同。

例 3.13 用 X=xx 形式打印出表达式的值。

```
10  INPUT "输入数据:";X,Y,Z
20 W =  SQR (X * X + Y * Y + Z * Z)
30  PRINT "X=";X;"  ";"Y=";Y;"  ";"Z=";Z
40  PRINT "W=";W


]RUN
输入数据:5,6,7
X=5  Y=6  Z=7
W=10.4880885
```

第 30 句的 PRINT 中,既有字符串表达式又有算术表达式。

3.3.2 分隔符“,”和“;”在 PRINT 中的用法

1. 分隔符“,”

在 PRINT 语句中,如果输出表达式中用了一个逗号,那么在输出时,逗号之后的表达式的值将占用 16 个字符位置。

例 3.14 求$\left(\frac{a}{b}+\frac{c \cdot d}{e}\right)^3$的值,其中 a=-123.78,b=1324,c=54.32,d=123.3,e=7.82

```
10  INPUT "输入数据:";A,B,C,D,E
20 F = (A / B + C * D / E) ^ 3
30  PRINT "A","B","C"
40  PRINT A,B,C
50  PRINT "D","E","F"
60  PRINT D,E,F

]RUN
输入数据:-123.78,1324,54.32,123.3,7.82
A               B               C
-123.78         1324            54.32
D               E               F
123.3           7.82            628067080
```

上例输出数据的排列格式是这样的:

字母 A 占据 1~16 列,字母 B 占据 17~32 列,字母 C 占据 33~40 列。同样,-123.78 占据新行上的 1~16 列,1324 占据 17~32 列,54.32 占据 33~40 列。其余类似。

在 PRINT 中,如果我们用逗号作为分隔符,那么系统按 16

位分为一组，将输出宽度为 40 个字符的屏幕分成三组，前两组各占 16 位，第三组占 8 位。如果输出的第一个表达式的值占满了 16 位，那么下一个表达式的输出值就占据第三组的 8 位。如果第三个输出值需要占据 7 位以上的字符位置，那么系统将把第三个输出值自动推到下行的第一个字符位置开始输出。这种格式称为标准输出格式。我们在准备输出数据时，应当注意数据位数不要超过 15 位。在第三组区输出时不要超过 7 个字符。

2. 分隔符“;”

在 PRINT 中，如果用“;”作分隔符，那么数据按紧凑格式输出。

例 3.15 按紧凑格式输出表达式的值。

```
10  INPUT "输入数据:";A,B,C,D,E
20 F = (A / B + C * D / E) ^ 3
30  PRINT A;B;C;D;E;F


]RUN
输入数据:-123.78,1324,54.32,123.3,7.82
-123.78132454.32123.37.82628067080
```

从上面结果可以看到:紧凑格式输出的数据之间没有空格，以致造成不大好辨认各个数据。如果在每两个数据之间输出一个空格，那么可读性就强多了。请读者自己完成。

3.3.3 打印空行和换行打印

1. 打印空行

如果输出数据每两行之间希望留有较宽的空白，那么可以在输出数据时有意打印一个空行。

例 3.16 打印空行

```
10  INPUT "输入数据:";A,B,C,D,E
20 F = (A / B + C * D / E) ^ 3
30  PRINT A,B,C
35  PRINT
40  PRINT D,E,F

]RUN
输入数据:-123.78,1324,54.32,123.3,7.82
-123.78           1324            54.32

123.3             7.82            628067080
```

上述程序中第35句打印的就是一个空行,其作用相当于把打印机的打印头向下移动了一行。因而在输出数据间留了一个空白行。也可以把35句改为:35 PRINT " ",效果完全一样。

2. 换行打印

如果需要把数据打印在不同的行上,可以有两种换行的办法。一是人为的安排换行打印;二是由机器自动换行。

例3.17　不换行打印

```
10  INPUT "输入数据:";A,B,C,D,E
20 F = (A / B + C * D / E) ^ 3
30  PRINT A;"  ";B;"  ";C;"  ";
40  PRINT D;"  ";E;"  ";F

]RUN
输入数据:-123.78,1324,54.32,123.3,7.82
-123.78  1324  54.32  123.3  7.82  628067080
```

在第30句中,由于C之后有一个分号,就表示打印完C的

值后不换行，第 40 句打印的值紧跟其后。尽管 30 和 40 句分别是两行上的两个打印语句，但它们打印的数据却安排在同一行上。因此，打印语句的个数和所占行数不能确定打印数据占用的行数。示例中的一个“␣”表示一个空格。有时为了准确描述打印格式，用“␣”表示一个空格，用“␣”明确表示出空格是个好办法。当然 C 之后也可以跟逗号，同样表示不换行，但打印格式就必须按逗号的要求办理。

实际上我们也可以不必人为换行，而由机器自动调整换行。

例 3.18　机器自动换行打印

```
10  INPUT "输入数据:";A,B,C,D,E
20 F = (A / B + C * D / E) ^ 3
30  PRINT A,B,C,E,F


]RUN
输入数据:-123.78,1324,54.32,123.3,7.82
-123.78          1324          54.32
123.3            7.82          628067080
```

当所有表达式的值在一行上打印不下时，机器自动转到下一个新行上继续打印。本例中第 30 句就是如此。

我们已经介绍了赋值语句、键盘输入语句和打印语句，利用这些语句就可以编写一个简单而完整的程序了。为加深读者对程序设计的理解，我们再举一个例子。

例 3.19　求：

$$\frac{e^{ax}\cos(2bx+c)}{2a}-\frac{e^{ax}[a\cos(2bx+c)+2b\sin(2bx+c)]}{2(a^2+4b^2)}$$的值。

其中：$a=3, b=4, c=5, x=6$

```
10  INPUT "输入数据:";A,B,C,X
20 W1 =  EXP (A * X)
30 W2 = 2 * B * X + C
40 W3 =  COS (W2)
50 W4 =  SIN (W2)
60 W = W1 * W3 / (2 * A) - (W1 * (A * W3 + 2 * B * W4)) / (2
 * (A * A + 4 * B * B))
70  PRINT A,B,C
80  PRINT X,"W=";W

]RUN
输入数据:3,4,5,6
3                    4                    5
6                    W=-10234605.6
```

这个程序中引入了一些中间工作变量,虽然直观性差了一些,但书写起来却更为方便。对初学者来说,我们首先要求编写程序正确,只要熟练,自然就能生巧。从上面给出的示例中,我们已经体会到编制程序的一些初步方法:

①熟悉问题,弄懂解题方法。如果问题的解决方法都不清楚,那无论如何是编不出程序来的。

②给变量以正确的命名。实际问题中给出的名字并不一定符合 BASIC 语言的要求。因而要对实际问题中的变量进行改造,如希腊字母或罗马文字母改用相近的英文或汉语拼音字符,使之合法化。

③重复出现的式子用中间工作变量代替,以备后用。

④变量用什么方式获得数据。这里仅介绍了 INPUT、GET 和赋值语句的办法,以后还会介绍另外的办法。总的原则是:简单和方便。

⑤数据的输出格式。总的原则是:数据格式要便于阅读和浏

览，尽量照顾人们通常的使用习惯。

3.4 BASIC 中常用的 DOS 命令

这里所谓的常用命令是操作系统提供给 BASIC 系统使用的一些命令。有些命令可以当成 BASIC 的语句来使用，而有些命令则只能作为键盘操作命令来使用。

1. 程序结束标志 END

格式：END

作用：使程序终止执行，系统不提供任何提示信息。

用法：该命令可以象 BASIC 语句一样写在程序需要终止的地方，表示程序执行到此完结。

例 3.20　带有 END 标志的程序。

```
10  INPUT "输入数据:";A,B,C
20 F = (A * A + B * B) / C
30  PRINT "F=";F
40  END


]RUN
输入数据:4,3,5
F=5
```

本章的所有示例程序的末尾均可以加上程序结束标志 END。

2. 运行程序命令 RUN

格式：RUN 或 RUN 〈行号〉

作用：运行内存中的 BASIC 程序。如果带有行号，则从指定行号开始运行。

用法:一个 BASIC 程序输入到内存以后,如果我们希望立即执行之,则在程序末尾敲入 RUN 命令,机器就按行号由小到大的顺序,立即执行输入到内存中的 BASIC 程序。RUN〈行号〉表示从指定行号的语句开始运行。

3. 列出程序命令 LIST

格式:LIST 或 LIST [〈行号 1〉][,〈行号 2〉]

作用:在屏幕上显示内存的 BASIC 程序。

用法:我们输入一个 BASIC 程序后,如果希望查看程序输入是否正确,那么就可以用该命令列出程序。如果不指出行号,则列出全部程序清单。如果指出了行号,那么就列出从行号 1 到行号 2 之间的程序段。如 LIST↙就显示出内存中保存的 BASIC 程序。

4. 删除程序的命令 NEW

格式: NEW

作用: 删去内存中的 BASIC 程序。

用法: 如果我们要输入一个新的程序,就应当把原来不用的程序删去。否则,新老程序就混在一起,影响运算的正确性。清除旧程序可以直接敲入 NEW 命令。如果你要检查旧程序是否已被删去,可用 LIST 命令。

5. 列磁盘目录

格式: CATALOG

作用:列出磁盘上文件的名字。

以上是经常用到的一些命令和操作,其它还有一些命令请参见第三册《操作系统》的有关节段。

3.5 BAISC 的功能键

在 CEC-BASIC 语言中,除了可以使用自身的语句和 DOS 提供的命令以外,还可以使用系统定义的功能键。恰当地使用这些功能键可以给我们的操作带来极大的方便。

3.5.1 用 CTRL 组成的功能键

将键盘上的 CTRL 键和某些字母键组合在一起可以组成新的控制键。

CTRL+C——停止执行 BASIC 程序返回到提示符"]"状态。如果运行程序中需要中途停止就可直接敲入这一功能键。CTRL+C 的敲键动作是:先压住 CTRL 键不松手,再敲 C 键。以下功能键的敲键动作与此类似。

CTRL+B——无条件返回到 BASIC 语言系统。如果目前系统处于 DOS 系统或监控程序之下,当需要回到 BASIC 状态时,则可敲入 CTRL+B。

CTRL+S——暂停程序的卷动。当用 LIST 列出较长的程序时,程序以卷动的方式显示在屏幕上,这不便于检查程序。当我们敲入 CTRL+S 后,程序就停止卷动而停留显示在屏幕上。当需要继续卷动显示时可敲入任何键。CTRL+S 功能键常常在用 LIST 浏览程序时使用。注意:CTRL+S 和 CTRL+C 两个功能键用法上的区别。

CTRL+X——将刚才输入的一行字符全部清除并把光标移到下一行的开头以便于重新输入。

再强调一下,凡是用 CTRL 键组合起来的功能键,在按键

时，必须先压住[CTRL]键不松手（松开后就不起作用），再敲入相应的组合键。

3.5.2 用 Esc 引导的功能键

Esc 也是键盘上的一个键，它和其它键组合起来又可得到一些新功能。[Esc]和[CTRL]在敲键动作上不同，只需先敲一下[Esc]键（不能一直按住），再敲其它键就行了。由 Esc 来引导的功能键有：Esc＋@——清除屏幕字符，光标定位到左上角。类似 Home 命令，Esc＋@的敲键动作是：先敲一下 Esc 键（必须放开），再敲@键。其余功能键的敲键动作与此类似。

Esc＋E——清除光标所在行右端的字符。

Esc＋F——清除光标右下角的字符。

Esc＋A——光标右移一个字符位置。

Esc＋B——光标左移一个字符位置。

Esc＋C——光标下移一行。

Esc＋D——光标上移一行。

在实际使用 Esc＋A、Esc＋B、Esc＋C 和 Esc＋D 功能键来控制光标的移动时不太方便，因为敲一次 Esc＋A，只能将光标右移一格，当要右移 5 格时就必须敲 5 次 Esc＋A。下面的控制键可以克服上述弱点，即敲一次 Esc 键，可以引导多次。如：敲一次 Esc 键，可多次使用 I、M、J 或 K 键。

Esc＋I——光标上移一行。

Esc＋M——光标下移一行。

Esc＋J——光标左移一个字符位置。

Esc＋K——光标右移一个字符位置。

3.5.3 箭头键◁和▷

除了上述的组合键以外,CEC－I 键盘上还有◁和▷两个三角形箭头键。

◁——将光标左移,并清除屏幕上箭头所经过的字符。注意:这些被清除的字符在没有敲↙之前屏幕上仍然能够看得见,但对系统来说,这些字符已不存在了。如果输入程序时发现以前输入的某个字符有误,那么可以用◁键将光标回退到出错字符处,再敲入正确字符。然后用▷向右移,一直移到先前输入字符的正确位置,再继续输入字符。

▷——将光标右移,并恢复先前被◁清除的字符。

3.6 BASIC 程序的删改

在输入 BASIC 程序时,即便是老手,敲键出错也在所难免,初学者就更容易发生这类错误。输入程序出错有两种情况:一是当敲入某个字符时发现该行先前某个字符有错。这时可用◁键将光标左移到出错处,敲入正确字符后,再用▷键向右退到后继字符的输入位置,继续敲入后继字符。(请考虑:为什么要用▷右退到后继字符的输入位置?)。二是已经换行后才发现字符敲键有误,那么只好用专门的修改方法来改正之。

3.6.1 程序的修改

1. 整行重新输入

如果一行程序中错误很多,或者出错行较短,那么我们可以将出错的行全部重新输入。那么系统就用刚输入的行去代替内存中原来的行。注意行号一定要相同。否则,系统误认为是添加

一个新行。

2. 修改一行中的个别字符。

如果待修改的字符个数相同，例如：仅仅是将 A 改成 B，或 XYZ 改成 ABC，那将是容易的。修改方法是：

①用 LIST 〈行号〉列出有误的语句行。

②将光标移到待修改的语句行的最左端（行号的第一个数字位置上）。

③用▷将光标右移到出错字符处并修改之。

④用▷将光标继续右移到该语句行的末尾（右端），再敲回车键。这个语句行就改好了。

例 3.21　修改下列程序。

```
10 INPUT A,B,C
20 F=(A*A+B*B)/C
30 PINT "F=";F
40 END
```

显然，程序中 30 句的 PINT 有错，我们利用上述方法修改如下：

①用 LIST 30 列出有误的程序行。（由于该例中程序较短，因而这个步骤可以省略。）

②用 Esc＋I 将光标上移到 30 语句行，再用 Esc＋J 将光标左移到 30 语句行的开头（注意不能用◁来左移）。该步操作后如下所示：

```
10 INPUT A,B,C
20 F=(A*A+B*B)/C
30 PINT "F=";F
40 END
```

③用▷将光标右移到"PINT"的"I"字符上，并输入"RI"两个字符。

④用▷将光标右移到最右端（即字母 F 之后），再敲回车

键。

用 LIST 列出程序：

```
10 INPUT A,B,C
20 F=(A*A+B*B)/C
30 PRINT "F=";F
40 END
```

由此可见，程序完全正确。

3. 删除多余字符

如果程序行中有多余的字符而影响到语句执行的正确，那么我们必须将多余字符删去。这可分两种情况：

一是多余字符不在引号内，则可以用修改个别字符的办法，将多余字符换成空格，因为空格并不影响程序的执行。

二是多余字符在引号内，则可用▷键将光标右移到出错字符处，再用 Esc＋A 或 Esc＋K 右移经过多余的字符，这就把多余字符删去了(注意：虽然屏幕上还能看见多余字符，但内存中它已不存在了)。最后，用▷将光标右移到该行最末尾，敲入回车键。

例 3.22　去掉下列程序中的多余字符。

```
10 INPUT A,B,C
20 F=(A*A+B*B)/C
30 PRINT "F+=";F
40 END
```

显然，30 句的引号内多了一个"＋"号，现将其删去。

①用 Esc＋I 将光标上移到 30 句，再用▷将光标右移到"＋"上。

②用 Esc＋A 将光标右移到"＝"上，再用▷把光标右移到末尾字母 F 后的空白位置上。

③敲入回车键。

经过这几步，多余字符"＋"被去掉了。再用列表命令列出。

```
LIST↙
10 INPUT A,B,C
20 F=(A*A+B*B)/C
30 PRINT "F=";F
40 END
```

4. 插入字符

如果要在一个语句的某个字符位置之前插入若干个字符，方法是：

①用 LIST 〈行号〉列出语句行。

②用 Esc＋I 和 Esc＋J 将光标移到该行的开头(最左端)。

③用▷将光标右移到待插入字符的位置之前。

④用 Esc＋I 将光标上移一行(列位置不变)。

⑤敲入待插字符(一个或多个均可)。

⑥用 Esc＋M 将光标下移一行，再用 Esc＋J(千万不能用◁)把光标左移到待插入位置的字符上。

⑦用▷将光标右移到最后的空白位置上，再敲入回车键，插入字符就算完成了。

例 3.23　在打印语句的左引号之前插入 A,B,C 使之成为：

```
10 INPUT A,B,C
20 F=(A*A+B*B)/C
30 PRINT A,B,C, "F=";F
40 END
```

插入的操作过程如下：

①用 LIST 30 列出原语句行为：

```
30 PRINT "F=";F
```

②用 Esc＋I 将光标上移到 30 行上。

③用▷将光标右移到"F="中引号位置(左引号)上。即：

```
30 PRINT "F=";F
```

④用 Esc+I 将光标上移到一个空行，列位置不变。

```
          _           (短横表示光标位置)
30 PRINT  "F=";F
```

⑤敲入 A ,B,C,后显示为：

```
          A,B,C,_        (短横代表光标)
30 PRINT"F=";F
```

⑥用 Esc+M 将光标下移到 30 行，再用 Esc+J 把光标左移到"F="的左引号上，即

```
          A,B,C
30 PRINT  "F=";F
```

⑦用▷将光标右移到 F 后的空白位置上，再敲回车键，这时需要的字符就插入了

```
LIST↙
10 INPUT A,B,C
20 F=(A*A+B*B)/C
30 PRINT A,B,C, "F=";F
40 END
```

3.6.2 程序行的插入

如果要在已有程序行之间插入一个或多个程序行，那么只需要将新的程序行号及其程序敲入就行了。

例 3.24 在例 3.23 的 20 和 30 之间插入一个 25 句。只需敲入：

```
25 PRINT "A=";A ,"B=";B ,"C=";C↙
LIST↙
10 INPUT A,B,C
```

```
20 F=(A*A+B*B)/C
25 PRINT "A=";A ,"B=";B ,"C=";C
30 PRINT "F=";F
40 END
```

由此可见,插入程序行是简单的。鉴于我们有可能在程序行之间插入新行,因而在行号的取法上切记不要使两行行号数值间隔为 1 连续。如行号为:20,21,22,23…等是不可取的。

3.6.3 程序行的删除

1. 删去一个整行。

如果需要去掉一个程序行,可以直接敲入该行的行号后再敲回车键,就把该行删去了。

例 3.25　删去例 3.24 中的 25 句

```
25↙
LIST↙
10 INPUT A,B,C
20 F=(A*A+B*B)/C
30 PRINT "F=";F
40 END。
```

2. 删去相邻的若干行

格式:DEL 〈行号 1〉,〈行号 2〉

作用:删去指定行号内的语句。

DEL 〈行号 1〉,〈行号 2〉——删去包括行号 1 和行号 2 之间的全部语句。

3. 删除全部程序行

欲删除全部程序行,最简单的办法就是敲入 NEW 命令。当然,也可以用上述的 DEL 命令,在此种情况下,行号 1 应该是程序的起始行行号,行号 2 是程序的最末行行号。

问题与思考

1. 哪些情况下用赋值语句使变量获得数据比用 INPUT 更方便?

2. INPUT 和 GET 语句的主要区别是什么?

3. 赋值语句和表达式有什么关系?

4. 逗号在哪些地方用到? 其作用如何?

5. 一个语句放在程序的任何位置上其运行结果均相同吗? 为什么? 举出一个例子。

习题三

1. 按格式写出下列程序运行的结果(为简便,每句后的↙被省略,下同)。

```
10 INPUT "请输入数据:";A,B,C,D,E,F,G,H
20 W=A+B+C+D+E+F+G+H
30 PRINT A,B,C,D,E,F,G,H
40 PRINT
50 PRINT "A ","B" ,"C","D"
60 PRINT A,B,C,D
70 PRINT "E" ,"F" ,"G" ,"H"
80 PRINT E,F,G,H
90 PRINT "A=" ;A,"B=" ;B,"C="; C
100 PRINT D;E;F;G;H
110 PRINT "W=";W
120 END
RUN↙
```

请输入数据：1.111111111，2.222，3.333，4.444，5.555，6.66，2，5↙

2. 指出下列程序的书写错误。

求 $Me^{-\beta t}\cos(\omega t+\psi)$ 的值，其中 $M=10$，$\beta=3.5$，$t=3$，$\omega=30°$，$\psi=5$。程序如下：

```
10 INPUT M,T
20 ω=30°
30 φ=5
40 H=M*E^(-β*T)*COS(ω*T+φ)
50 β=3.5
60 PRINT H,β
70 END
RUN
? 10 3.5
```

3. 编写下列问题的程序。

①求

$$\left(\frac{a-c}{a+b}+x\right)^2+\sqrt{y^2+4x^2\sin\beta x}\div e^4x^2$$

的值（自己给出所需数据）。

②按下列格式打印表格（编写出程序）。

姓名	性别	年龄	班级
张扬	男	15	初二.一
李芳	女	14	初二.二

……

第四章　转向和分支程序

在实际应用中，有时候希望跳过一些语句或根据某种条件来选择执行某些语句，这就要用到转向和条件分支语句。BASIC为达到这种目的，提供了转向语句和条件语句。

4.1　转向语句

格式：GOTO〈行号〉

作用：无条件转到指定行号的语句去执行。其中：GOTO是转向语句的标志，行号是程序中需要转去的语句行号。转向语句使我们能够多次地执行某些语句，也可以使我们跳过某些语句。

我们来看下面的例子（为了简洁起见，从本章起，语句末尾的回车键↙省写，但在输入程序时一定要敲入，切记）。

例4.1　分别给出三组A、B、C的值，求

$$(A+B^C-\frac{A}{BC})/(A^2+B^2)$$

的值。

```
5 I = 0
10  INPUT "输入数据:";A,B,C
20 I = I + 1
30 W = (A + B ^ C - A / B / C) / (A * A + B * B)
40  PRINT "第";I;"组的值"
50  PRINT "A=";A;"  ";"B=";B;"  ";"C=";C;"  ";"W=";W
```

```
60  GOTO 10
70  END

]RUN
输入数据:3,4,5
第 1 组的值
A=3  B=4  C=5  W=41.074
输入数据:
```

在这个例子中,首先输入第一组值,经过计算后分别输出A、B、C、W 的值。当执行到第 60 句时,又转回到第 10 句,要求输入第二组值。如此重复,直到三组值输入完毕。但由于 60 句是一个无条件转向语句,因而尽管三组值已输完,但程序仍然要求再输入数据。这时,如果希望程序停止执行,只好敲 CTRL+C 键,使程序强行中断回到“]”提示状态。

程序比较简单。其中的变量 I,用于统计输入了几组值。

4.2 条件语句

对于例 4.1 的程序,我们已指出它是不能自动停止执行的(除非我们不输入数据),原因就在于使用了一个无条件转向语句 GOTO 10,导致它老是转向第 10 句。但现实使用中,总希望执行若干次以后能够自动停下来,因而需要用到条件语句。

4.2.1 条件语句的格式和作用

1. IF-GOTO 型条件转向语句

格式:IF 〈条件〉 GOTO 〈行号〉

作用：如果条件成立，就把控制转到指定的行号去执行。否则，执行该语句的后继语句。

所谓条件成立与否，是这个意思：

如果条件是关系表达式或逻辑表达式，只要表达式的值为真，那么就称条件成立，否则称为条件不成立（即关系表达式的值为假）。

如果条件是算术表达式，只要算术表达式的值不为零（若表达式的绝对值小于或等于 2.9387E－39，则作为零处理），就称条件成立，否则称为条件不成立（即算术表达式的值等于零）。

如果条件是字符串表达式，只要字符串表达式的值不是空串，就表示条件成立，否则就表示条件不成立（即字符串表达式的值为空格）。

这里对条件成立与否的解释，适用于整个 CEC-BASIC 的条件。

我们用例子来说明条件语句的用法。

例 4.2　为便于对照，仍用例 4.1 为例，说明 IF 中条件是关系表达式的情况：

```
5 I = 0
10  INPUT "输入数据:";A,B,C
20 I = I + 1
30 W = (A + B ^ C - A / B / C) / (A * A + B * B)
40  PRINT "第";I;"组的值"
50  PRINT "A=";A;"  ";"B=";B;"  ";"C=";C;"  ";"W=";W
60  IF I < 3 GOTO 10
70  PRINT "程序结束"
80  END
```

```
]RUN
输入数据:3,4,5
第 1 组的值
A=3   B=4   C=5   W=41.074
输入数据:7,8,9
第 2 组的值
A=7   B=8   C=9   W=1187767.57
输入数据:1.35,3.24,4.78
第 3 组的值
A=1.35   B=3.24   C=4.78   W=22.4789377
```

在第 60 句,如果 I<3 为真,就再转 到第 10 句输入一组值并执行计算和输出。当 I=3 时(实际已输入了三组值),条件不成立,就自动执行第 60 句的后继语句 70 PRINT “程序结束”,打印出一个程序结束标志,提示用户,需要输入计算的三组数据已经完成。这时系统返回到“]”提示状态。

请读者将该例中的条件分别改为算术表达式和字符串表达式后再试之,并考虑三种表达式的用法有什么不同。

2. IF-THEN 型条件语句

格式:IF 〈条件〉 THEN 〈行号〉

作用:如果条件成立,则执行 THEN 之后指定行号的语句。否则,顺序执行该语句的后继语句。

这里的条件,其含义同前述,它可以是关系表达式、逻辑表达式、算术表达式或字符串表达式。行号必须是程序中出现的行号。这种形式的条件语句同 IF 〈条件〉 GOTO 〈行号〉 形式的条件语句用法完全相同。在前面例 4.2 的例子中,只要把程序中

的 IF-GOTO 换成 IF-THEN 均能正确运行。请读者自己完成。

3. IF-THEN〈语句〉型条件语句

格式：IF〈条件〉THEN〈语句序列〉

作用：如果条件成立，则执行 THEN 之后给定的语句序列，如果没有转向语句，执行完给定语句序列后顺序执行后继语句。如果条件不成立，就直接执行后继语句。

这里的语句序列可以是一个或多个语句。若是多个语句，语句彼此之间用冒号“：”隔开。

例 4.3　求 W 的值

$$W=\begin{cases}\frac{X}{a}e^{\frac{-x^2}{2a^2}}\cdot \mathrm{Sinx} & x>0 \text{ 且 } a\neq 0\\ \mathrm{COS}(ax+5) & \text{其它}\end{cases}$$

编出程序如下：

```
10  INPUT "输入数据:";A,X
20 W =  COS (A * X + 5)
30  IF A < > 0 AND X > 0 THEN W = X / A *  EXP ( - X * X /
(2 * A * A)) *  SIN (X)
40  PRINT "A=";A;"   ";"X=";X;"   ";"W=";W
50  PRINT "程序结束"
60  END

]RUN
输入数据:15,21
A=15  X=21  W=.439608606
程序结束
```

程序执行情况是这样的：

执行第 10 句首先输入 A、X 的值(本例中为 15 和 21)。因为 W 的取值只有两种情况,因而先假设一种条件满足,这就是第 20 句的作用。第 30 句再来判断给定条件是否成立。若条件成立,则 W 又送入了 X/A * EXP(−X * X/(2 * A * A)) * SIN(X)的值,这就把先前 20 句送入的假定值冲掉了。若条件不成立,那么 THEN 之后的赋值语句不执行,W 的值就是 20 句赋给的值,所以,第 40 句输出的 W 值总是正确的。

至此,有的读者会问,上例中的 THEN 之后是一个赋值语句,其后能否是 IF 语句呢?当然是可以的。IF 的最典型情况是:

IF 〈条件 1〉 THEN IF 〈条件 2〉 THEN…

4. 多路分支程序

格式:ON 〈n〉 GOTO 〈行号 1〉 [〈行号 2〉…〈行号 m〉]

作用:根据 n 的值,选择指定行号中的第 n 个行号的语句来执行。

这里的 n 称为条件变量,其值必须是正整数且 0⩽n⩽255。n 也可以是一个复杂的算术表达式。如果 n 的值为实数,那么机器自动转换成整数。如果 n 的值为零或者 n 值超界(即 n 所指的行号不在程序中),那么就执行后继语句。如果 n＜0,就发生语法出错。

例如:ON 3 GOTO 20,60,80,因为 n=3,所以应该转到第 80 句去执行。实际上,n 的值必须和行号的排列顺序号相对应。应该特别强调,这种多路分支,不是转到第 n 行去执行。

例 4.4 求 Y 的值。

$$Y=\begin{cases} A+BX^2+CX^3 & 0.5\leqslant T<1.5 \\ AC\cos(BX^2) & 1.5\leqslant T<2.5 \\ \sqrt{A+BX^3}+e^c & 2.5\leqslant T<3.5 \\ \ln\left|B+\dfrac{C}{X}\right| & 3.5\leqslant T<4.5 \end{cases}$$

要用 ON 形式的条件语句来解决这个问题，关键是构造出一个 n，使其值同 T 有关且刚好落到所需行号上。我们仔细观察就会发现，这里 T 的变化是有规律的。如果我们直接用 T 来代表 n，那么当 T=0.5 时就无法凑成 1，因而在 T 基础上加 0.5 再取整，即 n=INT(T+0.5)。试一试能否满足要求。在构造 n 值的同时，我们假定 Y 的四种情况依次对应 n 的 1，2，3，4。计算 Y 的表达式的语句行号在 ON 中的排列顺序号也要和 n 值对应。还是先来看下面的程序吧。

```
10  INPUT "输入数据:";T
20 A = 5:B = 6:C = 7:X = 8
30 N =  INT (T + 0.5)
40  ON N GOTO 60,80,100,120
50  PRINT "T值超界"
55  GOTO 140
60 Y = A + B * X * X + C * X ^ 3
70  GOTO 130
80 Y = A * C *  COS (B * X * X)
90  GOTO 130
100 Y =  SQR (A + B * X ^ 3) +  EXP (C)
110  GOTO 130
120 Y =  LOG ( ABS (B + C / X))
```

```
130  PRINT "T=";T;"   ";"Y=";Y
140  PRINT "程序结束"
150  END

]RUN
输入数据:1
T=1   Y=3973
程序结束
```

这里反复运行了四次，每种情况都执行了一次，我们给出一个 T 值，就得到相应的一个 N 值。在上例的第 40 句中，若

N＝1，相当于 GOTO 60；

N＝2，相当于 GOTO 80；

N＝3，相当于 GOTO 100；

N＝4，相当于 GOTO 120。

对于有规律变化的量用 ON 多路分支语句还是比较方便的。但是，对于无规律或者不容易找出规律的问题，要使用 ON 就麻烦一些了。这种情况有两种方法可以解决。一是重复书写 ON 中的行号，如 ON n GOTO 60,60,80,100,100 等。请看下面的例子。另一种实现方法见例 4.6。

例 4.5　问题同例 4.4，只是条件改变了。

$$Y=\begin{cases} A+BX^2+CX^3 & 0\leqslant T<2 \\ AC\cos(BX^2) & 2\leqslant T<3 \\ \sqrt{A+BX^3}+e^c & 3\leqslant T<5 \\ \ln\left|B+\dfrac{C}{X}\right| & 5\leqslant T<8 \end{cases}$$

```
10  INPUT "输入数据:";T
20 A = 5:B = 6:C = 7:X = 8
30 N =  INT (T + 1)
40  ON N GOTO 60,60,80,100,100,120,120,120
50  PRINT "T值超界"
55  GOTO 140
60 Y = A + B * X * X + C * X ^ 3
70  GOTO 130
80 Y = A * C *  COS (B * X * X)
90  GOTO 130
100 Y =  SQR (A + B * X ^ 3) +  EXP (C)
110  GOTO 130
120 Y =  LOG ( ABS (B + C / X))
130  PRINT "T=";T;"  ";"Y=";Y
140  PRINT "程序结束"
150  END

]RUN
输入数据:3.5
T=3.5  Y=1152.10387
程序结束
```

第 40 句中相同的行号 60,100,120 重复了多次,为的是使条件变量 N 的值和行号相对应。对于上述问题,也可用下面的办法来实现,

例 4.6　用多个 IF 实现 ON 多路分支。

```
10  INPUT "输入数据:";T
20 A = 5:B = 6:C = 7:X = 8
30  IF T < 2 AND T >  = 0 THEN 60
35  IF T >  = 2 AND T < 3 THEN 80
40  IF T >  = 3 AND T < 5 THEN 100
45  IF T >  = 5 AND T < 8 THEN 120
50  PRINT "T值超界"
55  GOTO 140
60 Y = A + B * X * X + C * X ^ 3
70  GOTO 130
80 Y = A * C *  COS (B * X * X)
90  GOTO 130
100 Y =  SQR (A + B * X ^ 3) +  EXP (C)
110  GOTO 130
120 Y =  LOG ( ABS (B + C / X))
130  PRINT "T=";T;"  ";"Y=";Y
140  PRINT "程序结束"
150  END

]RUN
输入数据:1.5
T=1.5  Y=3973
程序结束
```

请读者自己试一试其它值，运行结果同例 4.5 是否一样。

实际上，用多个 IF 来代替 ON 的功能有时更为简单，因为不必挖空心思去构造条件变量 n 同语句行号的对应关系。

不论是转向语句还是条件语句，语句中要求的行号都必须

是程序中能够提供的行号，否则就会发生错误。

4.2.2 条件语句的执行流程

前面详细介绍了条件语句的用法，为清楚起见，下面给出它们各自的执行流程图。

1. IF 〈条件〉 GOTO 〈行号〉

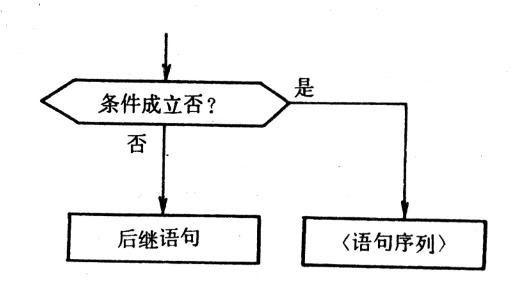


图 4.1 IF—GOTO 执行流程图

2. IF 〈条件〉 THEN 〈行号〉

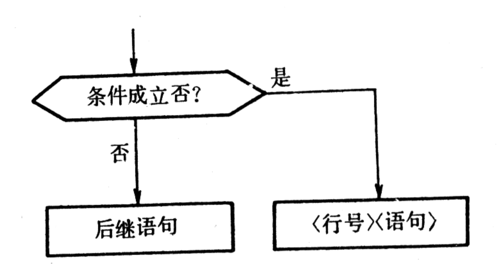


图 4.2 IF—THEN 执行流程图

3. IF 〈条件〉 THEN 〈语句序列〉

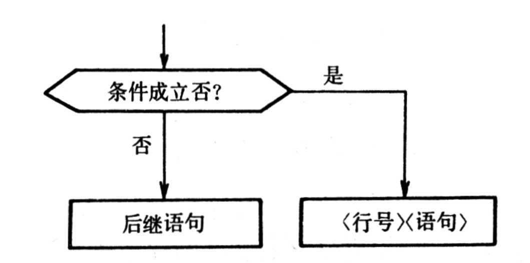


图 4.3 IF—THEN〈语句序列〉执行流程图

4. ON n GOTO〈行号 1〉,〈行号 2〉…〈行号 m〉

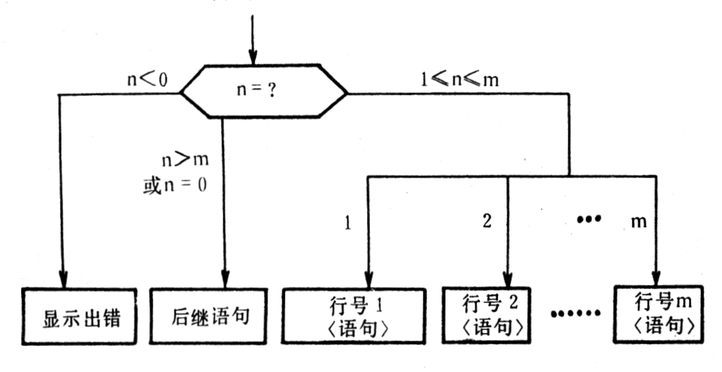


图 4.4 ON n GOTO 执行流程图

从以上四个流程图,我们就可以清楚地看见条件语句的执行流程。

4.3 读数据和置数据语句

由前面的示例中已经发现,为使变量获得数据,几乎都采用 INPUT 语句。虽然这种输入数据的方式既简单又方便,但如果我们要重新执行一次程序,那么仍然得重新从键盘上敲入值,即

使先前已经输入过的数据，也必须如此。有什么办法能使我们仅敲入一次数据而可以多次重新运算程序呢？读数据和置数据语句是最好的帮手。

4.3.1 读数据、置数据的格式和作用

格式：READ 〈变量表〉

作用：将 DATA 语句提供在数据区的数据依次送入到指定变量名中。

这里的变量表是一个或多个变量名，如果有多个变量名，变量名彼此之间要用“，”隔开。

格式：DATA 〈数据表〉

作用：为机器内部开辟的数据区提供数据。

数据表是一个或多个数据。如果是多个数据，数据彼此之间必须用“，”隔开。这里所说的数据只能是数值常数或字符串常数，不能是变量名、函数或带运算符的表达式。

如果一个程序中使用了 READ 语句，那么程序中一定要有 DATA 语句。

我们举例说明 READ 和 DATA 语句的用法。

例 4.7　编写一个程序，它能够按如下形式统计和打印学生成绩。（保留一位小数）

学号 (X1)	姓名 (X2$)	数学 (X3)	语文 (X4)	物理 (X5)	化学 (X6)	总分 (W)
631	张扬	85	90	92	73	…
633	李力	93	86	90	91	…
635	王为	78	82	80	76	…
平均成绩	…	…	…	…	…	…

假设：X1——学号，X2＄——姓名，X3——数学，X4——语文，X5——物理，X6——化学，W——总分。S3——数学的和，S4——语文的和，S5——物理的和，S6——化学的和，S7——总分和，I——学生人数。

解决该问题的思路是：

①读入一个学生的学号、姓名以及各科成绩。

②将各单科成绩累加求和。

③求各个学生的各科总分。

④用 DATA 语句提供数据，用 READ 语句读入数据。

⑤按报表格式打印。

下面是程序清单：

```
5 S3 = 0:S4 = 0:S5 = 0:S6 = 0:S7 = 0
10 I = 0
15  PRINT "学号 ";"姓名 ";"数学 ";"语文 ";"物理 ";"化学 ";"
总分"
20 W = 0
25 I = I + 1
30  READ X1,X2$,X3,X4,X5,X6
40 W = X3 + X4 + X5 + X6
50 S3 = S3 + X3
60 S4 = S4 + X4
70 S5 = S5 + X5
80 S6 = S6 + X6
90 S7 = S7 + W
100  PRINT X1;" ";X2$;"  ";X3;"   ";X4;"   ";X5;"   ";X6;"
 ";W
110  IF I < > 3 THEN 20
113 S3 =  INT (S3 / I * 10) / 10
```

```
114 S4 =  INT (S4 / I * 10) / 10
115 S5 =  INT (S5 / I * 10) / 10
116 S6 =  INT (S6 / I * 10) / 10
117 S7 =  INT (S7 / I * 10) / 10
120  PRINT "平均成绩  ";S3;" ";S4;"    ";S5;" ";S6;"   ";S7
130  DATA  631,"张扬",85,90,92,73
135  DATA  633,"李力",93,86,90,91
140  DATA  635,"王为",78,82,80,76
150  END

]RUN
学号 姓名 数学 语文 物理 化学 总分
631 张扬  85   90   92   73   340
633 李力  93   86   90   91   360
635 王为  78   82   80   76   316
 平均成绩 85.3 86   87.3 80   338.6
```

这个程序中用到了 READ 语句和 DATA 语句。DATA 的作用就是把张扬、李力、王为三个学生的学号、姓名和成绩依次放在数据区中,当执行到第 30 句的 READ 语句时,就直接到数据区去取学生的这些数据。经过统计和计算处理,打印出报表。

READ 和 DATA 语句究竟是如何配合使用的呢?原来,机器在内存中开辟了一个区域,专门用来存放 DATA 语句提供的数据,称为数据区。按照 DATA 语句在程序中出现的先后顺序,依次将 DATA 语句提供的数据存放在数据区中。为了指出数据区中数据的位置,系统内部还专门设立了一个指针,指向数据区的第一个数据位置,我们称为初始位置。随着数据的读取,指针可以自动向后推移。指针总是指向正要读取的数据。在本例中,数

据和指针的初始位置如图 4.5 所示。

当输入程序后，DATA 语句提供的数据在数据区内存放的情况如图 4.5(a)所示，指针指向当前值 631。这就是说，如果要读入的话，首先读入的就是 631。当第 30 句执行以后，就把张扬的数据取到了相应的变量中。这时指针已指向 633 这个值了(准备下一次读取)，如图 4.5(b)所示。第二次执行 READ 语句时，李力的全部数据又读入到相应的变量中，然后把指针指向 635。如此重复，直到把三个学生的数据读完。当读取了王为的化学成绩 76 以后，指针就指向数据区末尾的一个空单元，如图 4.5(c)所示。如果这时继续执行 READ 语句，就会出现数据不够用的错误。因为指针指向空单元就表示已经没有数据可供读取了。数据区和数据指针把 READ 和 DATA 语句联系起来了。

从以上可知，READ 并不是直接在 DATA 语句位置去读取数据，因而 DATA 在语句中的位置是无所谓的，可在 READ 之前，也可在其后。

READ 和 DATA 语句的优点是当我们再次用 RUN 重新运行程序时不必再输入数据，因为数据已写入了程序中，而每次用 RUN 重新运行时，系统总是把指针指向初始位置。缺点也是显然的。由于数据均要写入程序中，因而增加了程序的长度。

4.3.2 使用 READ 和 DATA 语句时的注意事项

1. READ 和 DATA 语句必须联合使用，只要程序中用到了 READ 语句，不管用了几次，也不管书写了几个，至少应有一个 DATA 语句与之对应。由于 READ 只能从数据区读取数据，因而必须有 DATA 向数据区提供数据，也只有 DATA 才能向数据区提供数据。

2. READ 语句中变量的个数、类型、顺序必须和 DATA 语

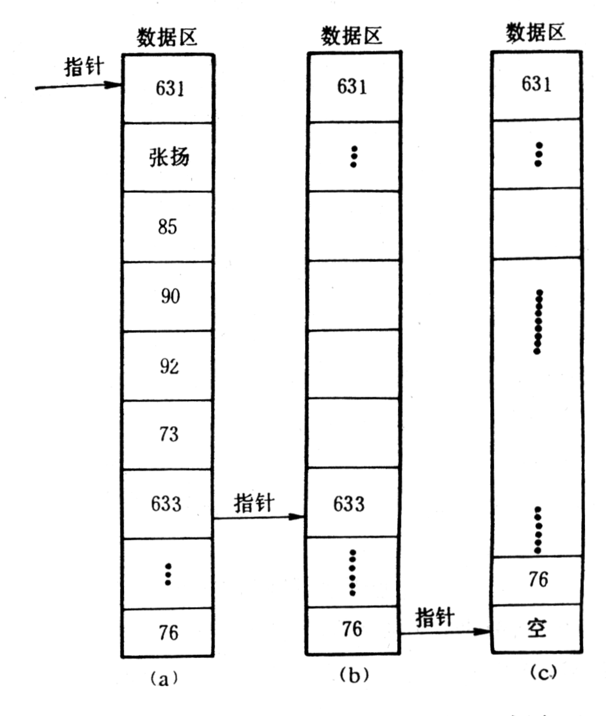


图 4.5 READ 和 DATA 执行示意图

语句提供的数据相对应。

一般来说，READ 语句中有几个变量名，DATA 语句就必须提供几个数据(当然可以多于变量的个数)。如果变量名多而数据少，则有些变量就得不到值。因而发生读入数据出错。

3. 一个程序中可以有一个很长的 DATA 语句(即数据多)，也可以把一个长的 DATA 语句拆成几个短的 DATA 语句(即数

据少一些)。如例 4.7 中的 130、135、140 语句可以合并成一个 DATA 语句,也可以把它们再分小一些。通常把相对独立的数据放在同一个 DATA 之后,便于检查和修改。如例 4.7 那样,每个学生的数据用一个 DATA 语句。DATA 的个数不影响数据区。但程序中各个 DATA 的先后顺序对数据区数据顺序有影响。

4. DATA 之后只能是常数。包括数值常数(可正可负)和字符串常数,且数据之间必须用“,”隔开。

5. DATA 在程序中的位置是任意的。但不同行号的两个 DATA 的先后顺序是重要的,因为它们的先后顺序决定了其提供的数据在数据区内的先后顺序。同样地,同一个 DATA 之后的数据排列的先后也是重要的,这个顺序取决于它和 READ 语句中变量名的对应关系。

4.4 数据指针的恢复

我们已经知道,每从数据区读取一个数据,数据指针就向下移动一个位置。当把数据区中数据读完以后,指针就指向数据区末尾后的一个空单元,这时如果继续读数据就会发生错误。如果我们再从头开始读取数据呢?当然是可以的。但必须设法把指针拔回到数据区的开始位置。所谓数据指针的恢复,就是把数据指针拨回到开始位置,使之指向第一个数据。如图 4.5(c),指针已经在数据区的末尾了,恢复指针就是使它重新指向第一个数据 631,恢复到图 4.5(a)的状态。恢复数据指针用到的语句是 RESTORE。

格式:RESTORE

作用:将数据区中的数据指针拨回到数据区的初始位置(指向第一个数据),以便 READ 语句重新读取数据区的数据。

例 4.8 求下式的值。

$$Y=\sqrt{A_1+A_2+A_3+A_4+A_5}+\sqrt[3]{B_1+B_2+B_3+B_4+B_5}+\sqrt[4]{C_1+C_2+C_3+C_4+C_5}$$

其中： A_1,A_2,A_3,A_4,A_5 分别取值为 11.42,135,72.4,58.3,66.4。

B_1,B_2,B_3,B_4,B_5 分别取值为 11.42,135,72.4,58.3,66.4。

C_1,C_2,C_3,C_4,C_5 分别取值为 11.42,135,72.4,58.3,66.4。

显然,三组值对应相同。对这个问题的程序可以用多种办法来实现。下面用 RESTORE 恢复数据区的办法来解决该问题。

```
10  READ A1,A2,A3,A4,A5
15  RESTORE
20  READ B1,B2,B3,B4,B5
25  RESTORE
30  READ C1,C2,C3,C4,C5
40 W1 = A1 + A2 + A3 + A4 + A5
50 W2 = B1 + B2 + B3 + B4 + B5
60 W3 = C1 + C2 + C3 + C4 + C5
70 Y =  SQR (W1) + W2 ^ (1 / 3) + W3 ^ (1 / 4)
80  PRINT "Y=";Y
90  DATA  11.42,135,72.4,58.3,66.4
100  PRINT "程序结束"
110  END
```

```
]RUN
Y=29.8429754
程序结束
```

程序的执行情况是：当 10 句读入 A_1,A_2,A_3,A_4,A_5 以后，指针已指向数据区末尾空白单元了。通过执行第 15 句的 RESTORE 语句，就把指针拨回到第 1 个数据 11.42 的位置。当第 20 句读入 B_1,B_2,B_3,B_4,B_5 时，指针又依次下推。如此重复。巧妙的使用 RESTORE 语句，就可以使我们能够重复读取数据区的数据。对于这个特定的问题，还有更简洁的办法，请读者自已考虑。

在使用 RESTORE 语句时需要注意：

①RESTORE 只能把数据指针拨回到数据区的初始位置，而不能拨到中间某个任意的数据位置。

②只有当需要从头读取数据时才需要用 RESTORE 来恢复指针。

③哪个 READ 语句需要从数据区的第一个数据位置重新开始读入，就在那个 READ 语句执行之前给出一个 RESTORE 语句(在程序中 RESTORE 的位置不一定必须放在 READ 之前，但其执行必须在 READ 之前)。

④如果恢复指针后某些位置上的数据并不需用，可以用无用的变量名占据相应的数据位置。如我们仅需要恢复 B_3,B_4,B_5 和 C_3,C_4,C_5 可用如下程序：

例 4.9　恢复读取部分数据。

```
10  READ A1,A2,A3,A4,A5
15  RESTORE
```

```
20  READ T,T,B3,B4,B5
25  RESTORE
30  READ T,T,C3,C4,C5
40  PRINT A1;" ";A2;" ";A3;" ";A4;" ";A5
50  PRINT B3;"  ";B4;"  ";B5
55  PRINT C3;"  ";C4;"  ";C5
90  DATA  11.42,135,72.4,58.3,66.4
100  PRINT "程序结束"
110  END

]RUN
11.42 135 72.4 58.3 66.4
72.4  58.3  66.4
72.4  58.3  66.4
程序结束
```

这里的 T 就是无用变量,它纯粹就是用来占据 11.42 和 135 两个数据位置的。

⑤需要特别强调的是:RESTORE 不是使控制返回到程序中的第一个 READ 语句去重新读入数据。RESTORE 只管恢复数据指针,至于是否用 READ 去读入,则由用户根据问题需要来确定。

4.5 框图法

在程序设计中,如果问题比较简单,那么可以直接编写程序。但是,实际应用中的问题往往比较复杂,需要考虑的因素比较多,程序的逻辑结构也很复杂,因而面对问题直接编写程序就会觉得千头万绪无从下手。为此,我们引入程序设计中的框图

法。所谓框图法，就是把解决问题的步骤用方框图的形式表示出来，然后再根据方框图来编写程序。一般说来，解决问题的框图画出来以后，根据框图来编写程序就不困难了。

1.框图法的表示符号

①用菱形框或半圆框来表示判断条件，框内写出条件名称。如：条件成立否？或条件成立否？

②用长方框来表示需要进行的操作，框内写出操作名称。如：A⇒B

③用箭头标出逻辑流向。通常用箭头把条件框和操作框联系起来。如图 4.6 所示。



图 4.6 框图联系示意图

一个实际问题不论有多复杂，均可用这种框图描述出来。根据用户的水平和需要，框图画得可粗（大概步骤）可细（详细步骤）。

2. 框图的例子

例 4.10 学生作息时间的框图。

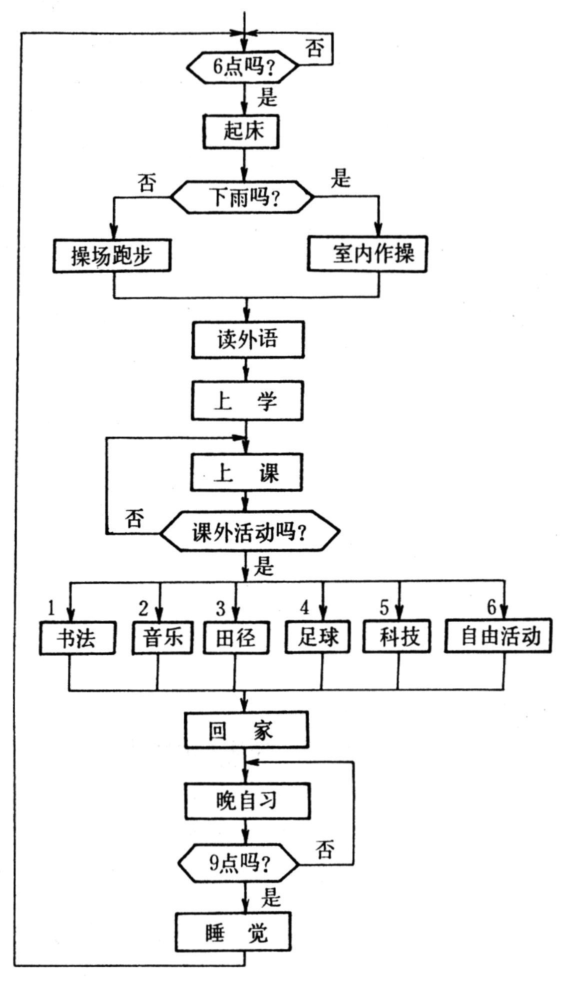


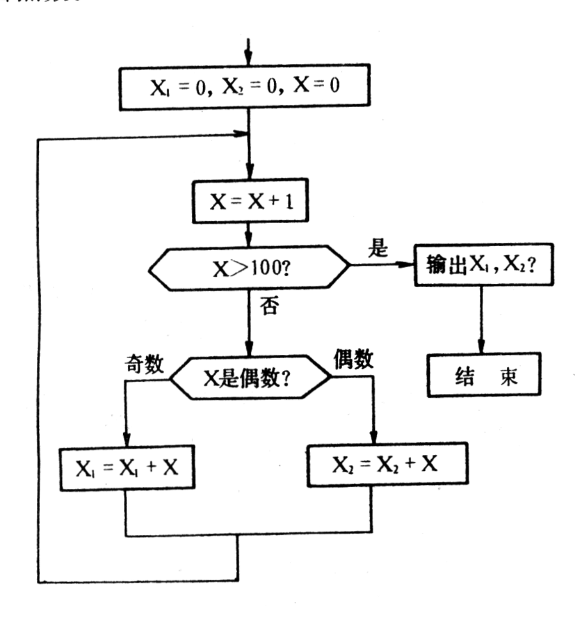
图 4.7 作息框图

实际上,我们每个人每天都在按一定的框图生活、工作和学习。

图 4.7 的框图还是比较粗的，它只描绘了一天的活动轮廓。我们还可以把每天的作息时间详细画出来。参照这个例子，请读者自己完成。

4.6 分支程序实例

例 4.11 求 100 以内所有奇数之和、偶数之和。
假设用 X_1 代表奇数之和，X_2 代表偶数之和，X 代表 1～100 之间的自然数。




```
10 X1 = 0:X2 = 0:X = 0
20 X = X + 1
```

```
25  PRINT X
30  IF X > 1000 GOTO 90
40  IF X =  INT (X / 2) * 2 THEN 70
50 X1 = X1 + X
60  GOTO 20
70 X2 = X2 + X
80  GOTO 20
90  PRINT "奇数之和=";X1;"    ";"偶数之和=";X2
100  PRINT "程序结束"
110  END

]RUN
奇数之和=250000    偶数之和=250500
程序结束
```

由于程序运行时间较长，为了不致太寂莫，第 25 句不断的输出 1,2,3…100 的自然数。

例 4.12 编写一个程序，它能够统计学生成绩并按分数段打印下列表格：

分数段	100	90～99	80～89	70～79	60～69	60 以下
人数	…	…	…	…	…	…
百分比%	…	…	…	…	…	…

假设分别用 G_{100}、G_{90}、G_{80}、G_{70}、G_{60}、G_{59} 代表各分数段的人数，用 G 来代表学生的成绩。X_{100}、X_{90}、X_{80}、X_{70}、X_{60}、X_{59} 代表各分数段所占人数百分比。I 代表学生人数。解题思路是：

①读入一个成绩，就判断其所属分数段。

②求各分数段所占人数百分比。

③按表格要求输出。

这个问题比较简单，我们直接给出程序：

```
5 I = 0:G100 = 0:G90 = 0:G80 = 0
8 G70 = 0:G60 = 0:G59 = 0
10  READ G
20  IF G =  - 1 THEN 90
25 I = I + 1
30  IF G = 100 THEN G100 = G100 + 1: GOTO 10
40  IF G >  = 90 AND G <  = 99 THEN G90 = G90 + 1: GOTO 10
50  IF G >  = 80 AND G <  = 89 THEN G80 = G80 + 1: GOTO 10
60  IF G >  = 70 AND G <  = 79 THEN G70 = G70 + 1: GOTO 10
70  IF G >  = 60 AND G <  = 69 THEN G60 = G60 + 1: GOTO 10
80 G59 = G59 + 1: GOTO 10
90 X100 =  INT (G100 / I * 100 * 10) / 10
95 X59 =  INT (G59 / I * 100 * 10) / 10
96 X60 =  INT (G60 / I * 100 * 10) / 10
97 X70 =  INT (G70 / I * 100 * 10) / 10
98 X80 =  INT (G80 / I * 100 * 10) / 10
99 X90 =  INT (G90 / I * 100 * 10) / 10
100  PRINT "分数段 ";"100 90-99 80-89 70-79 60-69 0-59"
110  PRINT "人数    ";G100;"     ";G90;"      ";
115  PRINT G80;"      ";G70;"      ";G60;"      ";G59
120  PRINT "百分比 ";X100;" ";X90;"  ";X80;"  ";
125  PRINT X70;"  ";X60;"  ";X59
130  DATA  80,85,91,93,76,84,66,79,74,78,63,62,88,100,42
135  DATA  74,86,82,78,95,77,84,66,78,94,85,67,53,-1
140  END
```

说明：

①第 135 句末尾的 －1 不是学生成绩，而是人为安排的数据结束的标志，它用于第 20 句判断读入数据是否完结。数据读入结束标志可以是任何一个不同于 DATA 提供的其它数据。

②为使打印的表格美观整齐，在安排打印表格时，有关行列应对整齐，以免输出时数据错位。

③为使变量名和分数段相对应，本例中用了 G_{100}，X_{100} 等作变量名。构成变量名的字符超过了两个，但由于所有变量名的第二个字符均不相同，所以不会发生混乱。

4.7 程序的打印、存盘和存带

程序输入机内，经过调试、修改和运行，证明程序已经正确无误后，我们可以将程序打印在宽行纸上，或者保存在磁盘或磁带上。

4.7.1 程序及运行结果的打印

CEC- I 机上打印运行结果及其程序清单可分西文和中文两种方式。

1. 西文方式下打印程序清单及其运行结果

在西文方式下要打印程序清单或运行结果，首先应该连通打印机，命令格式为：

格式：PR＃1

作用：在西文方式下连接打印机。

例 4.13 在西文方式下连通打印机。

]PR＃1↙

]LIST↙（这时将把内存中的程序在宽行纸上打印出来）

如果不再打印，可以断开打印机。

格式：PR＃0

作用：在西文方式下断开打印机。

例 4.14　断开打印机。

]PR＃0↙

]LIST↙（程序仅在屏幕上显示）

注意：在打印程序之前必须打开打印机的电源。这里所说的“连通”和“断开”不是指连通或断开打印机的电源，而是指建立或切断 BASIC 系统同打印机之间的联系。打印程序的运行结果与上述操作相同。

2. 中文方式下打印程序清单及其运行结果

格式：POKE 1659,〈n〉

作用：在中文方式下连通打印机。其中 n 是打印的汉字字体。$0 \leqslant n \leqslant 15$。

若 $n=0$，则表示断开打印机。若 $1 \leqslant n \leqslant 15$，则表示打印时所用的字体（各号字体请参见第二册第三章）。

例 4.15　用 1 号字体来打印程序。

]POKE 1659,1↙

]LIST↙　　　　（按汉字方式打印出程序清单）

如果需要在不同的地方打印出不同字体的汉字，那么可以象 BASIC 语句一样将 POKE 1659,n 插到相应的位置上。

例 4.16　变换打印汉字的字体。

```
5  POKE 1659,1
10  PRINT "中华学习机"
15  POKE 1659,2
20  PRINT "中华学习机"
25  POKE 1659,3
```

```
30  PRINT "中华学习机"
40  END

]RUN
中华学习机
中华学习机
中华学习机
```

3. 设置打印行距

所谓打印行距就是打印的运行结果或程序清单上下行之间的距离。

格式:POKE 1915,〈n〉

作用:设置上下行之间的距离。n 的值为行间距。$0 \leqslant n \leqslant 255$。

通常 n 值可取为 8,上下两行间距就比较适中。如果 n 值太小,如 n=1,则两行之间距离太小,反而不太好看。当然,对打印的表格来说,上下行又希望不要隔得太远。

4. 设置每行打印的字符个数。

格式:POKE 2043,〈 n 〉

作用:设置每行能够打印的最多字符个数。$0 \leqslant n \leqslant 255$。

一旦设置了每行允许的最多字符个数,当一个语句行所容纳的字符个数超出这个最大数时,系统就自动换行。如果没有明确设定字符数,那么系统约定每行打印至多 40 个字符。这里的字符数量以西文字符计算的。如果打印的是汉字,那么一个汉字占两个字符位置。

5. 设置字间距

所谓字间距就是指前后两个字符之间的距离。

格式:POKE 1787,〈n〉

作用：设置字间距。0≤n≤255。

如果没有明确给出字间距，系统约定为1，这可以满足通常应用的要求。

对于POKE命令，一旦被设置就一直作用下去，直到关机或重新设立为止。如果我们经常用到此命令，那么可以将所有POKE命令写成一个BASIC程序保存在磁盘上，当需用时再调入运行之。

4.7.2 程序的存盘和调用

程序除了可以在宽行纸上打印出来以外，还可以保存在磁盘上。一般说来，有用的程序最好都保存在磁盘上，以便再用或修改。但使用中注意：固化BASIC下存盘和调用命令无效。因而凡是要用存盘和调用命令，都必须事先用DOS 3.3，来启动系统。

1. 存盘命令

格式：SAVE 〈文件名〉

作用：将内存中的源程序用给定的名字保存到磁盘上。

文件名就是给整个源程序取的名字，它可以是汉字，也可以是字母开头的字母或数字的序列，其中可以嵌入下划线。

例4.17 将内存中的程序保存在磁盘上。

```
10  INPUT "输入数据：";A,B,C
20  F=(A*A+B*B)/C
30  PRINT "F=";F
40  END
    SAVE E4-17↙
]
```

这就将程序用E4-17文件名保存在磁盘上了。

2. 调用命令

格式:LOAD〈文件名〉

作用:将已存入磁盘的源程序文件调入内存。

文件调入内存后,可以对文件进行修改、列表或运行。

例 4.18　将先前存入的 E4-17 程序调入内存。

```
NEW ↙           (清除内存)
LOAD E4-17      (将 E4-17 文件调入内存)
LIST ↙
10 INPUT "输入数据:";A,B,C
20 F=(A*A+B*B)/C
30 PRINT "F=";F
40 END
RUN↙
输入数据:3,4,5↙
F=5
```

由上可以看出:有了存盘和调用命令,输入、运行或修改程序就相当方便了。

需要强调指出的是:SAVE 和 LOAD 命令只有在 DOS 3.3(或 DOS 3.4)启动下才有效。即是说,直接开机使用固化 BASIC 时 SAVE 和 LOAD 命令不起作用。这是使用中需要注意的。如果用户的程序准备存盘,那么事先请一定用 DOS 来启动系统。

4.7.3　程序的存带和调用

在中华学习 CEC-Ⅰ中,不仅可以使用打印机和磁盘驱动器,还可以使用录音机的磁带来保存程序。

1. 存带命令

格式: SAVE [〈"文件名"〉]

作用:将内存的程序保存到录音机的磁带上。

这里的文件名必须用引号括起来。具体操作方法是：

敲入 SAVE 和文件名字以后，同时按下磁带机上的“PLAY”键和“REC”键，然后再敲计算机键盘上的回车键↙。待程序装入磁带后，计算机主机发出两次“嘟嘟”声，并在屏幕上显示出 BASIC 系统的状态提示符号“]”，这时按下磁带机的“STOP”键。这样就把内存中的 BASIC 程序保存到磁带上了。

如果使用该命令时不给出文件名，同样也可以把程序用文件的形式保存到磁带上。但以后用 LOAD 命令来调用程序文件时，系统就只能按文件在磁带上的位置来装入程序。

2. 调用磁带文件命令

格式：LOAD [〈“文件名”〉]

作用：将磁带上的程序文件调入内存。

把录音机上磁带内的文件调入内存的操作方法是：

在键盘上敲入 LOAD 和文件名字以后，按下磁带机的“PLAY”键，再敲键盘上的回车键↙。文件调入计算机内存后机器发出两次“嘟嘟”声，并在屏幕上显示刚才调入的文件名字和 BASIC 提示符“]”。如果装入的文件不是给定的文件名时，屏幕显示已装入的文件名并给出一个反相显示的字每“N”，且继续调入后继文件，直到装入给定的文件为止。如果 LOAD 命令中省略了文件名，那么表示装入磁带上第一个文件。

磁盘和磁带上都可以保存文件且所用命令均相同。两者的区别在于：存盘及其调用都比较简单和快速；存带及其调用较慢。另外，存带和调用文件名一定要用引号括起来，而磁盘中的文件名绝不能用引号括起来。

问题与思考

1．ON n GOTO 行号 1，行号 2，…，行号 m 形式的语句不是必需的，为什么？

2．为什么 DATA 语句可以放在程序的任何位置？

3．举一个 RESTORE 语句放在 READ 语句之后而又能达到恢复目的的例子。

习题四

1．指出下列程序运行的结果，并指出数据指针处在什么位置。

①
```
10  READ A,B,C
20  A=A+B+C
30  B=A+B+C
40  C=A+B+C
50  IF C>A+B+C THEN 20
60  PRINT "A=";A,"B=";B,"C=";C
70  DATA 1,2,3
80  END
    RUN↙
```

②
```
10  READ A,B,C,D,E,F
20  DATA 1,2,3
30  A=A+E+F
40  DATA 4,5,6
50  G=G+B+C+D
60  DATA 7,8,9,10,11,12,13
```

```
70   RESTORE
80   IF G<30 THEN 10
90   PRINT A,B,C,D,E,F,G
100   END
     RUN↙
```

2. 先画框图再编写程序。

$$i=\begin{cases}\frac{V}{WL}e^{\alpha t}\sin wt & R^2-\frac{4L}{C}<0\\ \frac{V}{L}te^{\alpha t} & R^2-\frac{4L}{C}=0\\ k_1e^{S_1t}+k_2e^{S_2t} & R^2-\frac{4L}{C}>0\end{cases}$$

其中：$W=\sqrt{\frac{1}{LC}-\frac{R^2}{4L^2}}$，　　$\alpha=\frac{-R}{2L}$

$$K_1=\frac{-V}{2L\sqrt{\frac{R^2}{4L^2}-\frac{1}{LC}}},K_2=-K_1$$

$$S_1=-\frac{R}{2L}-\sqrt{\frac{R^2}{4L^2}-\frac{1}{LC}}$$

$$S_2=-\frac{R}{2L}+\sqrt{\frac{R^2}{4L^2}-\frac{1}{LC}}$$

假定 R,L,C,V 为已知常数，求 $t=0,0.1,0.2,\cdots 10$ 时 i 的值各是多少。

3. 用 ON 多路分支编写程序。

$$F(t)=\begin{cases}\sqrt{t^2+1} & 0<t<1\\ t^2-1 & 1\leqslant t<2\\ t^2-2t & 2\leqslant t<3\\ t^2-4t & 3\leqslant t<4\end{cases}$$

4. 根据下列程序画出框图。

```
5   W=100
10  READ X
15  IF X=999 THEN 70
20  W1=W
25  Y=0
30  W1=W1 * (1+X)
35  Y=Y+1
40  IF W1>2 * W THEN 50
45  GOTO 30
50  PRINT "X=";X,"Y=";Y
55  GOTO 10
60  DATA 0.05,0.07,0.09,0.11,0.13,999
70  END
```

5. 将 a,b,c 三个数按从小到大顺序排列。先画出框图,再写出程序。

6. 已知数列:$1,\frac{1}{2},\frac{1}{3},\frac{1}{4},\cdots\frac{1}{n},\cdots$。问小于 10^{-8} 的是第几项? 写出程序。

第五章　循环程序设计

在现实生活中，循环的的例子实在是太多了。如象我们生活的昨天、今天和明天，一天复一天，真可谓“明日复明日，明日何其多”。昨天、今天和明天，似乎天天都在重复，而天天都不完全一样。去年，今年，明年，年复一年，永远如此。

5.1　什么叫循环

在 BASIC 语言中，也有专门用于描述“明日复明日”的语句，这就是 BASIC 的循环语句。在程序设计中，经常需要反复执行一段程序，而在循环的执行过程中，程序的执行情况也在悄悄地发生变化。就象日复一日，年复一年，表面上看起来没有什么变化，但从人的年龄、知识、对财物的消耗来说都在发生变化一样。很难想象，生活中如果没有循环，世界将会变成什么样子。同样，如果 BASIC 语言（其它语言也如此）不允许或不能解决循环问题，那么很多问题就只能是“可想而知”了。例如，求 1＋2＋3＋…＋100000 的值，我们可以借助 99999 个加号“＋”将这些数字加起来，但实际上是行不通的。

事实上，BASIC 语言有两种方法可以用来解决循环问题。一是条件语句结合转向语句（这在前面已经介绍了）；二是用 BASIC 提供的 FOR-NEXT 循环语句。

5.2 循环语句

5.2.1 循环语句的格式和作用

格式:FOR〈实型算术变量〉=〈算术表达式 1〉TO〈算术表达式 2〉〔STEP〈算术表达式 3〉〕〈语句序列〉

NEXT〈实型算术变量〉

作用:反复执行 FOR 和 NEXT 之间的语句序列,直到实型算术变量的值超过算术表达式 2 的值为止,然后再顺序执行 NEXT 的后继语句。

其中:FOR——循环的开始标志。

〈实型算术变量〉——一个取实数值的变量名,又称循环控制变量。循环控制变量的值可以决定循环是否继续进行。

〈算术表达式 1〉——又称为循环控制变量的初值,即第一次循环时的取值。它可以是一个数值常数,也可以是一个带有各种算术运算符的复杂算术表达式。循环执行时,机器首先将初值计算出来,再赋给循环控制变量,作为第一次循环的开始值。

〈算术表达式 2〉——用于控制循环的终止 ,又称循环终限。如果循环控制变量的值超过了这个终限值,那么就停止循环而顺序执行 NEXT 的后继语句。如果循环控制变量的值没有超过终限值,则继续执行 FOR 和 NEXT 之间(即循环体)的语句序列。终限可以是一个具体的数值常数,也可以是一个算术表达式,但循环控制变量和终限进行超过比较时,总是按代数值来比较。

TO——终限的标志,即说明 TO 之后紧跟的是循环语句的终限。

〈算术表达式 3〉——称为循环控制变量的步长或增量,即表示每次循环后循环控制变量又要增加多少值后才又进行下一次循环。步长同样可以是一个具体的数值,也可以是一个算术表达式。步长可以是正数,也可以是负数。STEP 是步长的标志。〔STEP 〈算术表达式 3〉〕是一个待选项,根据需要进行取舍。如果步长为 1,则待选项可以不写,暗示循环控制变量每次递增 1。

〈语句序列〉——又称循环体,即需要反复执行的部分。它由一个或多个语句组成。当然,循环体中可以包含赋值语句、条件语句、转向语句、输入语句、打印语句,甚至还可以是循环语句。

〈NEXT〉——循环范围的结束标志。FOR 和 NEXT 构成了需要循环执行的部分。FOR 之前和 NEXT 之后均不属该循环的范围。通常,系统要求在 NEXT 之后的同行上紧跟一个循环控制变量名,以标志 NEXT 是哪一个循环语句的结束标志。

我们还是来看看下面的例子吧。

例 5.1　求 100 以内所有奇数的和。

```
10 S = 0
20  FOR I = 1 TO 100 STEP 2
30 S = S + I
40  NEXT I
50  PRINT "S=";S
60  END

]RUN
S=2500
```

在这个具体例子中,第 20 句中的 FOR 是循环的开始标志,

I 是循环控制变量，1 是 I 的初值，100 是 I 的终限，2 是步长。第 30 句的 S＝S＋I 是循环体，这里循环体仅有一个赋值语句。NEXT 表示循环范围到此为止。从 20～40 句一起构成 BASIC 中的一个循环语句。

这个循环语句的执行情况是：

①将初值 1 送到循环控制变量 I 中。

②执行循环体 S＝S＋I，即 S＝S＋1。

③将循环控制变量增加一个步长(2)，即 I＝I＋2，将 I 值同终限值 100 比较，若 I 超过(这里暂时理解为大于)终限值，就跳出循环后转④。否则，再转到②继续执行循环体。

④执行 NEXT I 的后继语句，即 PRINT "S＝";S。

显然，上例完成了 1＋3＋5＋7＋…＋99 的累加计算，结果存放在 S 中。

例 5.2　设 A＝3，B＝5，X＝0.5，步长为 1.5，直到 10 时，求

$$Y=\frac{AX^2+BCOSX}{AX^2+BX^3SINX}$$的值。

```
10 A = 3:B = 5:I = 0
20  FOR X = 0.5 TO 10 STEP 1.5
25 I = I + 1
30 W = A * X * X
40 Y = (W + B *  COS (X)) / (W + B * X ^ 3 *  SIN (X))
50  PRINT "I=";I;"   ";"X=";X;"   ";"Y=";Y
60  NEXT X
70  PRINT "程序结束"
80  END

]RUN
```

```
I=1  X=.5  Y=4.89492407
I=2  X=2  Y=.205062576
I=3  X=3.5  Y=-.834029171
I=4  X=5  Y=-.145745333
I=5  X=6.5  Y=.311825431
I=6  X=8  Y=.0701980001
I=7  X=9.5  Y=-5.16915494
```

在程序中，变量 I 不但用于统计 X、Y 的组数，也可以用来记录循环的次数。

仔细观察输出的数据，读者就会发现，X 取值 9.5 以后，程序就停止循环了，为什么没有取到终限值 10 呢？因为 9.5＋1.5＝11，这个值已经超过了终限值 10，因而跳出循环。由此可以得出一般结论：循环语句中循环控制变量的值不一定刚好取到终限值。我们为什么称为终限而不称为终值就是这个原因。

5.2.2 循环语句的执行流程

在不同型号机器上的循环语句，其循环效果是相同的。但循环语句在具体实现时，执行流程可能不相同。对循环的处理来说，有的机器是循环控制变量得到初值后就马上与终限值进行比较，如果初值一开始就超过终限，那么导致循环一次都不执行。而有的机器又是循环控制变量得到初值后立即执行一次循环，然后将步长累加到循环控制变量中后再与终限值进行比较，以确定是否进行下一次循环。这种处理方法导致循环至少要执行一次，而不管初值和终限值如何。前一种方法我们称为是先比较后确定循环，后一种方法称为先循环后比较。处理方法不同，

得到的循环的次数和结果就不一样。中华学习机 CEC-Ⅰ采用的是先执行后比较的循环处理方法，因而，不管循环控制变量的初值和终限如何，总是要执行一次循环的，是否跳出循环，只有第二次循环开始时才能确定。

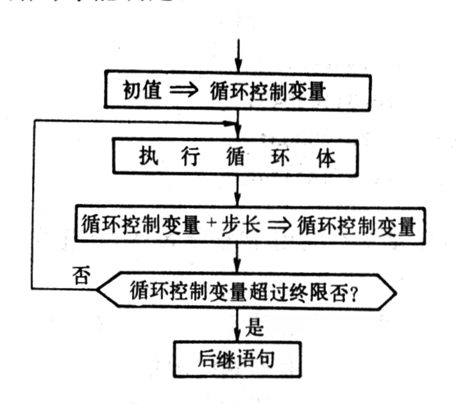


图 5.1　循环语句处理示意图

处理循环语句的执行流程如图 5.1 所示。

“循环控制变量超过终限否”中所谓“超过”的含义取决于步长的取值符号。

如果步长是正值(即增量)，那么“超过”的意思就是循环控制变量的值大于终限值；如果步长是负值(即减量)，那么“超过”的意思就是循环控制变量的值小于终限值。

5.2.3　使用循环语句时注意

1.　循环控制变量只能是实型变量，不能是整型变量或字符型变量，更不能是带有运算符的表达式。

例 5.3　求 1＋2＋3＋…＋100 的值。

```
10 S = 0
20  FOR I% = 1 TO 100
```

```
30 S = S + I%
40  NEXT I%
50  PRINT S;
60  END

]RUN

?Syntax error in 20
```

第20句发生语法错。原因就在于循环控制变量I%是整型变量。

2. 如果循环控制变量的初值一开始就超过终限值，那么循环也要执行一次（注意“超过”是什么意思）。

例5.4 我们用I来表示循环控制变量，用K来记录循环的次数，用S来代表累加和。

```
10 S = 0:K = 0
20  FOR I = 100 TO 50 STEP 2
30 K = K + 1
40 S = S + I
50  NEXT I
60  PRINT "K=";K;"  ";"S=";S;"  ";"I=";I
70  END

]RUN
K=1  S=100  I=102
```

从运行结果看，只循环了一次，因而K=1。累加和S=100，说明只把I的初值100加到S中去了。I=102说明把步长2累加到I中去了，因为初值为100，再累加上步长2，所以I=102。

3. 循环控制变量的值可以参与循环体中的运算，也可以只用于控制循环次数。但是切记：循环体中不能修改循环控制变量的值。如果修改了，就把正常的循环秩序打乱了，使得我们难以控制。

例 5.5 循环控制变量只用于控制循环次数。

```
10 W = 0
20  INPUT A
30  FOR I = 1 TO 100
40 W = W + A
50  NEXT I
60  PRINT "W=";W;"  ";"A=";A;"  ";"I=";I
70  END

]RUN
?4.72
W=472.000002  A=4.72  I=101
```

这里的 I 就只用于控制循环次数。如果在循环结束以后需要用到循环控制变量 I 的值，那么 I 的值就是最后那次循环的值(100)再加上步长(1)，因而输出的 I＝101。这是因为中华学习机采用先执行后比较的缘故。

4. 初值、终限、步长均可以是复杂的算术表达式。

例 5.6 初值、终限、步长是算术表达式的情况。

```
10 A = 5:W = 0:K = 0
20  FOR I =  SQR (A) TO  EXP (A / 3) STEP A + A
30 K = K + 1
40 W = W + I
50  NEXT I
```

```
60  PRINT "K=";K;"   ";"W=";W
70  END

]RUN
K=1  W=2.23606798
```

程序中K用于记录循环的次数，这里循环只执行了一次。因而K=1。

5. 步长可以是负值。

例5.7 步长为负值的情况。

```
10 S = 0
20  FOR I = 100 TO 1 STEP  - 1
30 S = S + I
40  NEXT I
50  PRINT "S=";S
60  END

]RUN
S=5050
```

如果步长为负值(如本例中的－1)，那么I是否超过终限1就用I<1来判断就行了。这个例子是把100,99,98,…,2,1累加起来。前面的求和是从小累加到大，这里是从大累加到小，实现方式不同，但效果完全一样。

6. 循环的次数可以用如下的公式来计算：

循环次数＝INT(ABS(终限－初值)/步长)＋1。

如果循环控制得不好，有可能丢掉一次应该执行的循环。

例5.8 下面的程序就少执行一次循环。

```
10 K = 0
20  FOR I = 0.5 TO 5 STEP 0.3
30 K = K + 1
40  PRINT "I=";I
50  NEXT I
60  PRINT "循环次数=";K
70  END

]RUN

I=4.7
循环次数=15
```

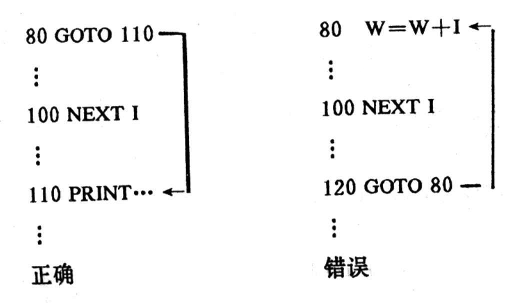
以上输出的是最后那次循环的结果。由此可以看到，I＝4.7时还应该再执行循环一次。但由于机内数据转换造成的误差，以致使本来应该循环的一次给丢掉了。当初值、步长为带小数时容易发生这种情况，避免丢掉循环的办法有二：

其一是在终限值上增加步长的一半。如上例终限取为5.15就不会丢失循环次数了。其二是将循环参数扩大若干倍，使初值、终限、步长均成为整数。

7. 在循环体内的转向语句可以转到循环体外，但循体外的转向语句不能转向循环体内。

例5.9 循环语句和转向语句的关系。

```
⋮                          ⋮
50 FOR I=1 TO 100          50 FOR I=1 TO 100
⋮                          ⋮
```



8. 在对话方式下使用循环语句时，FOR 和 NEXT 必须写在同一行上（以是否敲↙作为换行标志，只要敲入↙，就表示换到新的一行。所以纸上书写的一行和机器内部的行是不完全相同的。这里所说的一行是指用↙隔开的行）且字符个数（包括敲入的空格）不能超过 239 个。

9. FOR 和 NEXT 形式的循环语句只能用于解决循环次数已知的那一类循环。如果循环次数未知或根据某种条件来确定是否继续循环，那么就只能用条件语句和转向语句配合才能完成。

例如：求 $1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}+\cdots$，当$\frac{1}{n}<10^{-8}$时停止累加。这个问题不大好直接用 FOR-NEXT 循环语句来完成，因为当$\frac{1}{n}<10^{-8}$时需要循环多少次是无法事先知道的。这种情况只好用条件语句和转向语句来实现循环了。

5.3 多重循环

我们曾经提及，循环体中还可以包含循环语句，这样就构成所谓的多重循环。

5.3.1 多重循环的结构

多重循环的结构如下：

外循环：

FOR $I=I_1$ TO I_2 STEP J_3

⋮

内循环：

FOR $J=J_1$ TO J_2 STEP J_3

〈循环体〉

NEXT J

⋮

NEXT I

这是一个二重循环，它在循环控制变量为 I 的循环中又套了一重循环变量为 J 的循环。通常把 FOR J=…NEXT J 称为内循环，而把 FOR I=…NEXT I 称为外循环。作为外循环的循环体，除了整个内循环以外，还可以包含其它一些语句。

两重循环的执行情况如一个大轮子带动一个小轮子，大轮（相当于外循环）转动一圈，小轮（相当于内循环）就要转动若干圈。

当然，还可以有三重、四重循环。CEC-BASIC 至多允许嵌套 10 重循环。事实上，实际应用中很难有真正达到 10 重循环的情况。

例 5.10　两重循环的例子。

①A 赋给初值 1 后执行外循环的循环体。即 30～80 句。

②第 40 句是内循环的开始，它将初值 2 赋给 B，执行内循环的循环体。

```
10 W1 = 0:W2 = 0
20  FOR A = 1 TO 10
30 W1 = W1 + A
40  FOR B = 2 TO 10 STEP 2
50 W2 = W2 + B * W1
55  PRINT "W2=";W2
60  NEXT B
70 W3 = W1 / W2
80  PRINT "W3=";W3
90  NEXT A
100  PRINT "程序结束"
110  END

]RUN
W3=8.33333334E-03
程序结束
```

③当执行到 60 句时，由于 B 的值没有超过终限值 50，因而继续内循环，直到 B 超过终限后跳出内循环，执行其后继第 70 句。

④当执行到第 90 句时，由于 A 没有超过终限 10，因而重复外循环。至此，外循环执行了一次，而内循环却执行了 5 次。重复以上过程，直到循环结束。

从上述说明可知，外循环 A 执行一次，内循环 B 就执行 5 次。整个内循环共要执行 10×5=50 次。为清楚起见请参见图 5.2 所示。

5.3.2　使用多重循环时注意

在使用循环语句编写循环程序时，除了前面关于循环语句的注意事项以外，对于多重循环的情况，我们还要强调以下几点。

1.　中华学习机 CEC-Ⅰ允许循环语句嵌套的最大重数是10重。通常情况下用不到10重。

2.　内外循环不能互相交叉。一个循环要么全部包含在另一个循环之中，要么全部在循环之外。请参见图5.3所示

3.　多重循环嵌套可以共用一个出口NEXT，但必须由内到外逐层引出循环控制变量。

参见图5.3(b)所示。

4.　凡是与本层循环无关的计算提到循环语句之外，以减少重复计算，从而提高程序的运行速度。

5.　同一层循环放在同一列上。这样书写程序，层次关系清楚。

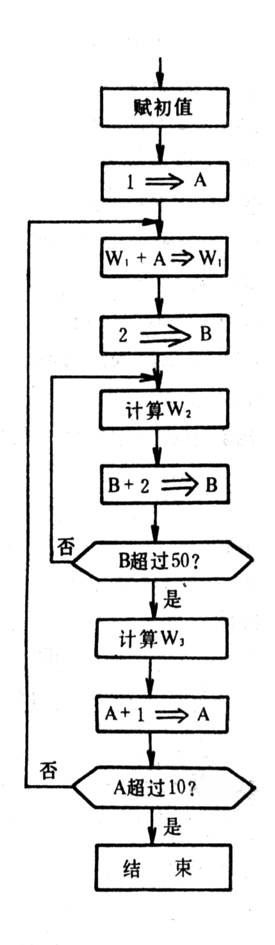
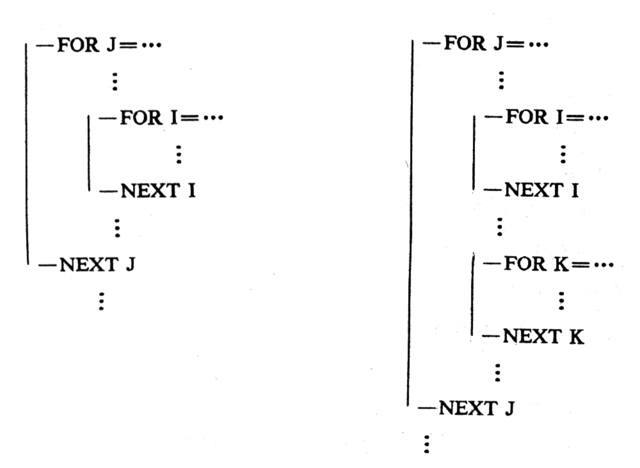
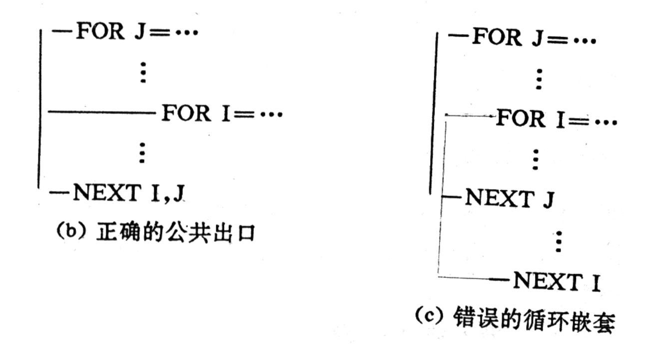


图5.2　例5.10框图



(a) 正确的循环嵌套



(b) 正确的公共出口

(c) 错误的循环嵌套

图 5.3　多重循环嵌套结构

5.4　循环程序实例

例 5.11　假设 $a_1, a_2, \cdots, a_{20}$ 为已知值，求多项式

$P(X) = a_1 + a_2x^2 + a_3x^3 + \cdots + a_{20}x^{20}$ 的值

```
5 PX = 0
10  INPUT X
15 W = 1
20  FOR I = 1 TO 20
30  READ A
35 W = W * X
40 PX = PX + A * W
50  NEXT I
60  PRINT "P(";X;")=";PX
70  DATA  1.32,4.56,7.88,5.67,3.77,6,2.8,3.11
75  DATA  7,2,3,3.33,5,6,1.66,3.45,2.44,7,5.5,5.678
80  END

]RUN
?1.5
P(1.5)=50908.4138
```

随着程序的执行,第35句就能形成X、X^2、X^3、…、X^{20}。

例5.12　设有$a_1,a_2,\cdots,a_{20}$共20个数,找出其中的最大数和最小数。

设G代表最大数,L代表最小数。

解决问题的思路是:将第一个数分别送到G和L中。再读入下一个数,如果这个数比G大,那么说明该数有可能成为最大数,因此,将该数送到G中。如果这个数小于G,那么再把该数同最小数L比较。如果该数比L还小,那说明该数有可能是最小数。否则,说明该数既不是最大数也不可能是最小数,因而被淘汰。再读入下一个数,重复以上过程,最终一定能够找出最

大数和最小数。框图如图 5.4 所示。

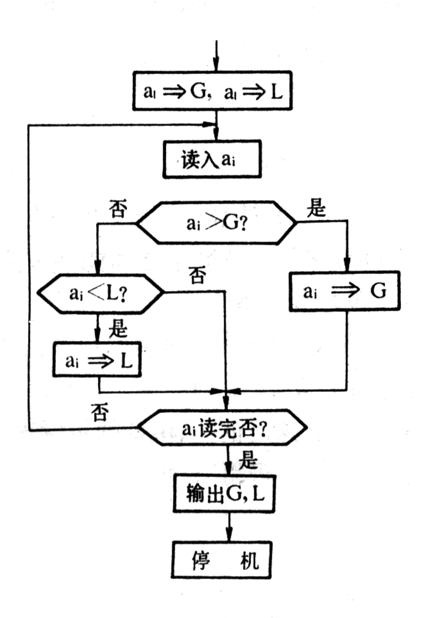


图 5.4　例 5.12 框图

```
10  READ A
20 G = A:L = A
30  FOR I = 1 TO 19
40  READ A
50  IF A > G THEN G = A: GOTO 70
60  IF A < L THEN L = A
70  NEXT I
80  PRINT "最大数=";G;"  ";"最小数=";L
90  DATA  12,34,23,56,45,78,123,567,34,88
```

```
95  DATA  77,78,56,90,46,76,32,10,56,99
100  END

]RUN
最大数=567   最小数=10
```

例 5.13 求一个三位的整数，其个位数字与十位数字之和能够被 2 整除；个位数字与百位数字之和能够被 3 整除；十位数字与百位数字之和能够被 4 整除。

我们用 A、B、C 分别代表所求整数的个位、十位和百位数字。而 A、B、C 分别可以取 1～9 中的某一个数字，如果这些数字能够满足所给条件，那么这时的 A、B、C 就是所求三位整数的个位、十位和百位。这个程序可用三重循环来实现。

```
10  FOR A = 1 TO 9
20  FOR B = 1 TO 9
30  FOR C = 1 TO 9
40  IF A + B =  INT ((A + B) / 2) * 2 AND A + C =  INT ((A +
 C) / 3) * 3 AND B + C =  INT ((B + C) / 4) * 4 THEN 60
50  NEXT C,B,A
55  PRINT "不存在满足条件的数": GOTO 75
60 W = C * 100 + B * 10 + A
70  PRINT "这个三位数是:";W
80  PRINT "程序结束"
90  END

]RUN
这个三位数是:531
程序结束
```

在该例中，如果给定的条件满足，就直接跳出循环而打印该

数据。这种不通过 NEXT 出口而中途跳出循环的情况称为循环的非正常出口。通过逐层循环的 NEXT 而依次退出循环的情况称为循环的正常出口。在该例中如果是正常出口，则表示不存在满足条件的数，因而第 55 句给出了找不到这种数的标志。

问题与思考

1. 用循环语句能够解决的问题用条件语句和转向语句也一定能够解决，反之亦然，这种说法正确吗？为什么？

2. 为什么转向语句不能转入循环体？如果强行转入会发生什么情况？

3. 转向语句经过 FOR 入口转入循环体行吗？为什么？

4. 循环语句的终限值肯定比初值大吗？

5. 循环控制变量的最后值一定是终限值吗？举例说明。

6. 为什么最好把与本层循环无关的量提到循环之外？

习题五

1. 指出下列程序的运行结果。

①
```
10 FOR A=1 TO 5 STEP 2
15 PRINT "A=";A;" ";
20 FOR B=A TO 5 STEP 2
25 PRINT "B=";B;" ";
30 FOR C=B TO 5 STEP 2
35 PRINT "C=";C;
40 NEXT C
50 NEXT B
60 NEXT A
```

```
70 END
RUN
```

②

```
10 FOR X=1 TO 3
20 FOR Y=1 TO 3
30 FOR Z=1 TO 3
40 IF X+Y+Z>5 THEN 95
50 IF X+Y+Z=5 THEN 85
60 IF X+Y+Z<>5 THEN 75
70 NEXT Z
75 PRINT X,Y,Z
80 NEXT Y
85 PRINT X,Y,Z
90 NEXT X
95 PRINT X,Y,Z
100 END
RUN
```

2. 指出程序的书写错误。

①

```
10 FOR A=1 TO 100
20 W=A*A*A
30 FOR B=1 TO 5
40 W=W+B
50 PRINT W
60 NEXT B
70 IF B〉100 THEN 40
80 GOTO 20
90 PRINT A
100 NEXT A
110 END
```

②

```
10 FOR A=1 TO 7
20 FOR B=2 TO 10 STEP 3
```

```
30 W=A*B+A/B
40 PRINT W
50 NEXT A
60 PRINT A
70 NEXT B
80 END
```

3. 已知 x_1、y_1，x_2、y_2，x_3、y_3，…，x_{20}、y_{20} 的值，求 $W = \sqrt{x_i - y_i}$ 的值。当 $x_i - y_i < 0$ 时打印出错标记，继续进行下一次求值运算。

4. 求 1000 以内能够除以 3 余 2 的整数，并统计这种整数的个数。

5. 编写 0°～180°之间每隔 1°的正弦、余弦、和弧度值表。并按如下格式输出：

三角函数表

角　度	弧　度	SIN	COS
0	…	…	…
1	…	…	…
2	…	…	…
⋮			
180	3.14159	…	…

6. 已知：$C_1, C_2, \cdots C_{10}$ 和 X 的值，求

$$C(x) = \sum_{i=1}^{10} (C_i x \prod_{j=1}^{4} \mathrm{Cos}\, jx)$$

的值。

7. 找出数字符号互不相同的一个四位整数，使得前两位数字之和的平方等于后两位数字之和的立方，如果没有这样的四位数，则给出说明信息。

第六章 数组和维语句

6.1 什么是数组

直到目前为止，程序中用到的所有变量，无论是数值型变量还是串型变量，它们均是以个体面目出现的。这犹如一个普通公民，他(她)既不代表某个单位，也不代表某级组织。但是又正如现实生活中一样，若干个体可以组成一个集体。例如：一个学生班级，一个学校，一个部门，一级组织等等都是一种典型的集体。在 BASIC 语言中，用于描述个体的是前面章节中介绍和使用的变量、常数和字符串。而用于描述这若干个体组成的集体则是数组。简单说来，数组就是若干同类数据所组成的集体。集体中单个成员，就叫数组的元素。全班学生的成绩数据可以组成一个数组，每个学生的成绩均是数组元素。而(5，10，25，30，40，80，100)和$\begin{pmatrix} 1 & 3 & 5 \\ 7 & 9 & 11 \\ 13 & 15 & 17 \end{pmatrix}$都可以用数组来表示。数组中包含的数据个数可以很多，也可以很少。对于数组我们有如下的要求：

1. 数组的名字

对任何相同类型的数据集合，如果我们需要作为数组的话，都必须给该集合取一个名字，这就是数组名。数组名同变量名的构成方法一样。

2. 分配存储空间

任何一个数组，一经说明，系统就必须在内存中为之分配存储空间。为了分配最适合的内存空间，说明数组时必须指出它所需要的最大存储空间。我们把这个最大的存储空间称为数组的大小。

3. 数组元素

定义数组是按一种集体的方式来说明的。但实际使用时却又是按个体形式来引用的。对数组中的个体我们称为数组元素，数组元素也是一种变量，称为下标变量。数组元素的位置序号是重要的，因为对数组元素的存取就是通过指出元素在数组中的位置序号来实现的。这种存取方法，犹如教师只须说出座位号（某行某列）就可以指定学生一样。

6.2 数组的表示方法

程序中如何表示一个数组呢？这是用维语句来说明的。

6.2.1 维语句的格式

格式：DIM 数组名 1(下标 1，下标 2，…)〔，数组名 2(下标 1，…)…〕

作用：定义一个或多个数组，并为数组在内存中预先分配存储空间。

其中：DIM 是数组说明符。数组名的构成方法同变量名一样。下标指明数组元素能够分配的最大存储空间，或者看成是数组元素能够变化的最大范围。下标的个数称为数组的维数。

1. 一维数组

能够用一个下标来描述的数组我们称为一维数组。例如：(50，25，37，19，82，73，95)数据集，可以用一维数组来描述。假定

该数据集的名字为 A，共有 7 个元素，用维语句说明如下：

DIM A(6)

由于分配空间总量从第 0 号开始分配，因此 A 的最大下标为 6。7 个元素的位置分别是：A(0)，A(1)，A(2)，A(3)，A(4)，A(5)，A(6)。这里的 A(0)，A(1)，A(2)，…A(6)就称为下标变量。

以上我们定义了一个一维数组 A。如果我们将 50 送到 A(0)，25 送到 A(1)，37 送到 A(2)，…，按这种顺序将(50，25，37，19，82，73，95)分别送到 A(0)～A(6)中，那么 A(4)的值就是 82，A(6)的值是 95。圆括号中的号码就是数组元素在数组中的位置。定义一个数组只是把存储位置固定下来了，但该位置具体是什么值，还要看我们送的是什么数据。犹如座位定了，但某个座位是张三吗还是李四，那就看具体是哪个人坐在那里了。直观说来，凡是排成一队的数据，我们都可以用一维数组来刻划之。

2. 二维数组

类似地，用两个下标来描述的数组我们称为二维数组。在实际生活中，凡是能够排成行和列(长方形、正方形)的数据我们都可以用二维数组来描述。如：

$$B=\begin{pmatrix}11 & 12 & 13 & 14\\ 21 & 22 & 23 & 24\\ 31 & 32 & 33 & 34\end{pmatrix}$$，用维语句描述如下：

DIM B(2,3)

数组 B 共有三行四列(包括第 0 行第 0 列)。这里的 B 是数组名字，2 和 3 都是下标。下标 2 表示数组 B 的行号变化范围是 0 到 2，下标 3 表示数组的列号变化范围是 0 到 3。

要表示数组 B 中的某一个元素，只能用下标变量的形式。如：B(0，1)表示数组 B 中第 0 行第 1 列的元素，B(1，3)表示数

组B中第1行第3列的元素。其余类似。数组说明只确定了元素的位置,但该位置究竟存放的是什么值,这同数组获得数据有关。由于下标变量仍然是一种变量,因而根据需要,我们可以给下标变量赋一定的值。如B(1,2)=23,B(2,3)=34等。

按照上述方法,我们还可以定义三维数组、四维数组等等。CEC-BASIC 至多可以定义到88维,但通常使用中一、二、三维数组最为普遍。

3. 下标和下标变量

在定义和引用数组元素时,下标和下标变量是两个不同而又有联系的概念。在维语句DIM中,下标用于刻划数组存储空间的大小,而在下标变量中,下标却用于确定元素在数组中的位置。

下标变量是指数组中的一个元素。如B(1,2),B(0,3)就是下标变量。下标变量中表现出来的下标可以是一个常数,也可以是算术表达式。如:B(1,2)中的下标1,2是具体的数值常数。如B(I+1,J)或B(I,J+1)中的I+1和J+1,均是算术表达式。

6.2.2 使用维语句时应注意

定义数组的方法是简单的。但实用中需要注意以下几点:

1. 数组一定要先说明后使用,程序中用到的数组,必须在存取之前用DIM来定义。如程序中需用到W数组,可作如下形式的说明:

```
10 DIM W(10,10)
```

这是一个11行11列的数组。如果一个数组最多11个元素(下标是10),那么这种数组可以不用DIM说明而直接使用。这种情况称为数组的隐式说明,直接用维语句DIM来描述的数组称为显式说明。

2. 用一个 DIM 说明符可以定义多个数组名，但各个数组名都必须给出各自的下标。如果 DIM 定义了多个数组名，那么各个数组名之间必须用逗号“,”隔开。如：

10 DIM A(3,5),B(10,8),C(10,10,10)

这里 A、B 是二维数组，C 是一个三维数组。A、B 和 B、C 之间用“,”隔开。

3. 如果一个数组是多维数组，那么描述维数的下标之间也必须用“,”隔开。如：

10 DIM W(10,20,30)

这是一个三维数组。10,20,30 之间均用“,”隔开各个不同的下标。

4. 下标允许的最大值是 32767，最小值是 0。数组的最大空间是下标值加 1 相乘的积。如：

10 DIM A(10),B(10,20),C(5,6,7)

数组 A 的最大空间＝(10＋1)＝11

数组 B 的最大空间＝(10＋1)×(20＋1)＝231

数组 C 的最大空间＝(5＋1)×(6＋1)×(7＋1)＝336

5. 在引用下标变量时不能超过下标范围。如：

10 DIM W(10,15)

下标变量 W(5,10)是正确的，而 W(11,3)或 W(5,18)均是不正确的，因为下标超界了。即是说，在 W 数组中找不到第 11 行 3 列和第 5 行 18 列这两个元素(位置)。

6. 在同一程序中，一个数组名只能用 DIM 说明一次，但可以使用多次。

例 6.1
```
10   DIM A(10,10),B(15,15)
…
50 DIM A (35,30)
…
```

这里的数组 A 定义了两次，这是不正确的。

7. 下标变量中的下标可以是一个具体的数值常数，也可以是一个算术表达式，而维语句 DIM 中的下标可以是正整数或已有值的变量名。

例 6.2

```
5 INPUT N,M
10 DIM A(N,M)
...
40 I=3:J=5
50 A(I+1,J+2)=153
...
```

这里 I+1 和 J+2 就是算术表达式。

8. 在定义整数组和字符串数组时，必须在数组名之后给出整数或字符串的标志。如：

10 DIM X＄(10,10),M%(5,5)

这里的 X＄是字符串数组，其数组中的元素只能取串型值。M%是整数组，数组元素只能是整型数值。

6.2.3 如何给数组赋值

用 DIM 定义数组，只是给数组分配存储空间，但如何把具体的数据传送到相应的位置呢？一般可用循环语句来实现。

1. 一维数组如何得到数据

假设有一个一维数组 A=(10,30,50,15,80,28)，用程序来描述如下例：

例 6.3 向一维数组传送数据。

```
10  DIM A(5)
20  FOR I = 0 TO 5
30  READ A(I)
```

```
40  NEXT I
50  PRINT "A(4)=";A(4)
60  DATA  10,30,50,15,80,28
70  END

]RUN
A(4)=80
```

第10句定义一维数组A，它只有6个元素。第20～40句对数组A读入数据。一重循环结束后就把数据区提供的数据分别送到了A的各个元素位置上。第30句中的A(I)是一个下标变量，随着I值的变化(0～5)，就提供了数组A的各个位置，因而能够把数据送到数组的相应位置。第50句打印出下标变量A(4)的值，以验证所需数据是否传送到相应位置上。

例6.4　生成数组B＝(10,20,30,40,50,60,70,80,90,100)

```
10  DIM B(9)
20  FOR I = 0 TO 9
30 B(I) = I * 10 + 10
40  NEXT I
50  PRINT "B(0)=";B(0);"  ";"B(8)=";B(8)
60  END

]RUN
B(0)=10  B(8)=90
```

由于数组元素的值是有规律的，所以就不必再用数据区的办法来提供数据了。随着循环语句的执行，第30句的B(I)＝I＊10＋10就能自动生成10,20,30,…,100的数据并送到相应

位置。

2. 二维数组如何得到数据

设有二维数组：

$$W=\begin{pmatrix} 35 & 72 & 48 & 63 & 94 \\ 101 & 32 & 49 & 53 & 68 \\ 11 & 75 & 77 & 69 & 61 \\ 54 & 47 & 98 & 78 & 87 \end{pmatrix}$$

显然这是一个四行五列的二维数组。

例 6.5　按行将数据送到 W 的各个元素位置。

```
10  DIM W(3,4)
20  FOR I = 0 TO 3
30  FOR J = 0 TO 4
40  READ W(I,J)
50  NEXT J,I
60  PRINT "W(3,1)=";W(3,1)
70  DATA  35,72,48,63,94
72  DATA  101,32,49,53,68
74  DATA  11,75,77,69,61
76  DATA  54,47,98,78,87
80  END

]RUN
W(3,1)=47
```

上述程序中，把行的变化作为外循环(I)，列的变化作为内循环(J)，这种输入方式的实际效果就是按一行一行的顺序，依次把数据送到数组 W 中。当然我们也可以按一列一列的顺序来

输入数据。按什么方式(按行或按列)输入数据,数据就应该按什么方式来排列。最好是按行输入,这比较符合我们的阅读和书写习惯。总之,数据要传送到正确的数组位置。

例 6.6 用程序来生成一个单位阵。

$$M=\begin{bmatrix}1&0&0&0&0&0\\0&1&0&0&0&0\\0&0&1&0&0&0\\0&0&0&1&0&0\\0&0&0&0&1&0\\0&0&0&0&0&1\end{bmatrix}$$

这个数组中的元素值是有规律的,除主对角线元素值为 1 以外,其余元素值均为 0。

```
10  DIM M(5,5)
20  FOR I = 0 TO 5
30  FOR J = 0 TO 5
40  IF I = J THEN M(I,J) = 1: GOTO 55
50 M(I,J) = 0
55  PRINT M(I,J);" ";
60  NEXT J
65  PRINT
70  NEXT I
80  END

]RUN
1 0 0 0 0 0
0 1 0 0 0 0
0 0 1 0 0 0
```

```
0 0 0 1 0 0
0 0 0 0 1 0
0 0 0 0 0 1
```

在程序中,随着二重循环的执行,40 和 50 两句就能使主对角元素值为 1,非主对角元素值为 0。其它任何维数的数组均可按上述的原则来定义和生成。

6.3 数组的应用

在程序设计中,数组用得相当广泛。有些问题必须用数组才能解决。而有些问题可用数组也可以不用数组。

例 6.7 用数组来统计学生成绩。

学号	语文	政治	数学	物理	化学	外语	生物	总分
611	85	87	86	83	79	77	78	…
614	75	79	80	76	77	79	66	…
……								

假设全班共有 10 人,总分由计算机自动统计,我们只需按要求输入原始数据,统计结果由计算机输出。解题思路是:

①用二维数组来描述学生成绩表。把学号和总分等计算在内共计 9 项(如上表)。

②用循环语句把原始数据输入到数组,机器自动统计总分并将分数送到总分的相应位置。

③按表格形式输出数据。

程序清单如下:

```
10  DIM S(10,9)
12  PRINT "学号"; TAB( 5);"语文"; TAB( 9);"政治"; TAB( 13);"
数学"; TAB( 17);"物理";
14  PRINT  TAB( 21);"化学"; TAB( 25);"外语"; TAB( 29);"生物"
; TAB( 33);"总分"
16  PRINT
20  FOR I = 1 TO 10
25 W = 0
30  FOR J = 1 TO 8
40  READ S(I,J)
50  IF J = 1 THEN  PRINT S(I,J);: GOTO 80
60  PRINT  TAB( J * 4 - 3);S(I,J);
70 W = W + S(I,J)
80  NEXT J
90 S(I,9) = W
100  PRINT  TAB( 33);W
110  NEXT I
120  DATA  611,85,87,86,83,79,77,78
121  DATA  614,75,79,80,76,77,79,66
122  DATA  616,77,75,73,80,68,79,85
123  DATA  618,90,93,96,92,88,89,90
124  DATA  622,66,64,68,75,74,77,65
125  DATA  624,64,63,58,68,70,72,54
126  DATA  630,57,58,64,65,66,63,61
127  DATA  635,80,78,79,77,75,74,73
128  DATA  636,88,89,90,87,86,83,90
129  DATA  642,77,74,76,78,80,71,68
130  END
```

```
]RUN
学号语文政治数字物理化学外语生物总分

611 85  87  86  83  79  77  78  575
614 75  79  80  76  77  79  66  532
616 77  75  73  80  68  79  85  537
618 90  93  96  92  88  89  90  638
622 66  64  68  75  74  77  65  489
624 64  63  58  68  70  72  54  449
630 57  58  64  65  66  63  61  434
635 80  78  79  77  75  74  73  536
636 88  89  90  87  86  83  90  613
642 77  74  76  78  80  71  68  524
```

请读者自己分析程序的执行情况。

例 6.8　将 n 个元素按从小到大的顺序排列。

这里必须要用到数组，解题思路是：

①定义和生成一维数组 A。

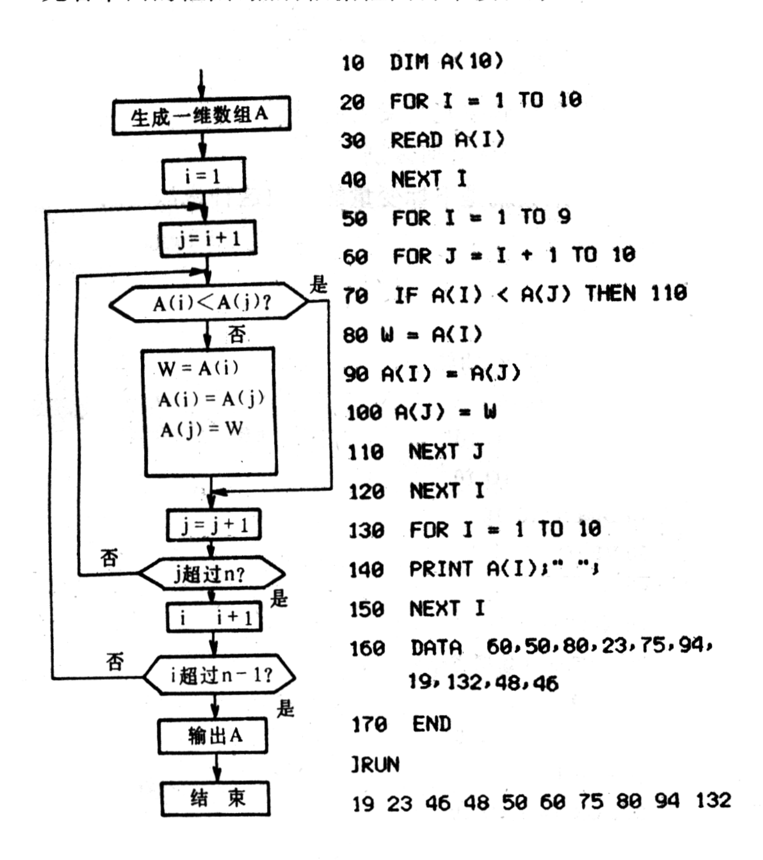
②通过比较和位置交换，把最小的数据移到数组的最前面，较大的数据移到数组的后面。

③通过 n－1 次比较可以得出一个最小值，再经过 n－2 次比较又可得到一个次小值。如此进行，直到把所有数据按从小到大排列为止。

④尽管处理的是一维数组，但必须用到二重循环。外循环控制将最小值移到数组的前面，其循环一次只能确定一个较小值。内循环控制使两个数据进行比较。内循环结束后才能确定一个

值是否较小。

先看下面的框图，然后根据框图再来读程序。




```
10  DIM A(10)
20  FOR I = 1 TO 10
30  READ A(I)
40  NEXT I
50  FOR I = 1 TO 9
60  FOR J = I + 1 TO 10
70  IF A(I) < A(J) THEN 110
80 W = A(I)
90 A(I) = A(J)
100 A(J) = W
110  NEXT J
120  NEXT I
130  FOR I = 1 TO 10
140  PRINT A(I);" ";
150  NEXT I
160  DATA  60,50,80,23,75,94,
     19,132,48,46
170  END
]RUN
19 23 46 48 50 60 75 80 94 132
```

说明；

20～40 句用于生成数组 A。

50～120 句用于将数组 A 的元素值排序。I 循环一次，就确定出一个较小值。

130～150 句用于输出排序后数组的值。

问题与思考

1. 为什么数组必须先定义后使用？
2. 下标、下标变量有何区别？
3. 数组的维数就是下标变量的个数这种说法对吗？

习题六

1. 指出下列程序的错误。

①
```
10 DIM A(10,10)
20 FOR I=1 TO 10
30 FOR J=1 TO 10
40 DIM A(15,15)
50 READ A(I,J)
60 NEXT J
70 NEXT I
80 DATA…(数据省略)
90 END
```

②
```
10 DIM A$(10,10)
20 FOR I=1 TO 10
30 FOR J=1 TO 10
40 READ A$(I,J)
50 NEXT J
60 NEXT I
70 DATA 10,20,30,80,90,100,75,85,95,…
80 END
```

2. 指出下列程序运行的结果。

```
10 DIM B(5,5)
20 FOR I=1 TO 5
30 FOR J=1 TO 5
40 IF I-J>=0 THEN A(I,J)=1:GOTO 60
50 A(I,J)=I-J+1
60 PRINT A(I,J);"  ";
70 NEXT J
80 PRINT
90 NEXT I
100 END
RUN↙
```

3. 用程序生成如下形式的矩阵。

① $\begin{vmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{vmatrix}$ ② $\begin{vmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{vmatrix}$

4. 将下列两个矩阵的对应元素进行交换。

$A=\begin{vmatrix} 1 & 3 & 5 \\ 7 & 9 & 11 \\ 13 & 15 & 17 \end{vmatrix} \quad B=\begin{vmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \\ 14 & 16 & 18 \end{vmatrix}$

5. 将卜列两个矩阵相乘。

$A=\begin{vmatrix} 15 & 24 & 31 \\ 52 & 28 & 19 \\ 11 & 14 & 17 \end{vmatrix} \quad B=\begin{vmatrix} 24 & 28 & 29 \\ 31 & 42 & 15 \\ 7 & 9 & 2 \end{vmatrix}$

第七章　BASIC 函数

根据函数的作用和运算结果，我们把 BASIC 函数分成如下几类：算术运算函数，字符串处理函数，数据类型转换函数，格式打印控制函数以及自定义函数。

7.1　算术运算函数

算术运算函数可以参与 BASIC 算术表达式中的算术运算或单独进行运算以得出一个数值。

7.1.1　三角函数

1. 求正弦值

格式：SIN(X)

作用：求 X 弧度的正弦值。

例 7.1　　? SIN(5)↙

　　　　　-.958924274

注意：①变量 X 可以是任何合法的算术表达式。

　　　②变量 X 必须用弧度来表示。

2. 求余弦值

格式：COS (X)

作用：求 X 弧度的余弦值。

例 7.2　　? COS (5)↙

　　　　　.283662186

3. 求正切值

格式:TAN(X)

作用:求 X 弧度的正切值。

例 7.3　　? TAN(5)↙

-3.380515

4. 求反正切值

格式:ATN(X)

作用:求 X 的反正切值。

例 7.4　　? ATN(5)↙

1.37340077

注意:变量 X 必须在$-\frac{\pi}{2}\sim+\frac{\pi}{2}$之间。

对于三角函数,我们要强调以下两点:

①提供的实在参数(调用时的参数)必须是弧度,如果实际应用中是角度,那么必须把角度转换成弧度。按 $180°=\pi$,$1°=\frac{\pi}{180}$公式进行转换。最好由机器自动进行转换。

②实在参数可以是数值常数、数值变量或任何合法的算术表达式。

7.1.2 其它数学函数

1. 求绝对值

格式:ABS(X)

作用:求 X 的绝对值。

例 7.5　　? ABS(-3)↙

3

2. 求平方根

格式:SQR(X)

作用:求 X 的平方根。

例 7.6　　? SQR(100)↙

10

注意:X 的值只能是正数。

3. 求 e^x 的值

格式:EXP(X)

作用:求 e^x 的值。

例 7.7　　? EXP(3)↙

20.0855369

4. 求自然对数值

格式:LOG(X)

作用:求 X 的自然对数 Lg(x)的值。

例 7.8　　? LOG(3)↙

1.09861229

注意:X 的值必须大于 0。

5. 取整数

格式:INT(X)

作用:取最接近 X 而又不大于 X 的最大整数。

例 7.9　　? INT(3.14159)↙

3

? INT(5.99998)↙

5

? INT(-3.14159)↙

-4

6. 求符号值

格式:SGN(X)

作用:取 X 的符号值,取值原则是:

$$若\ SGN(X)=\begin{cases}1,则\ X>0\\0,则\ X=0\\-1,则\ X<0\end{cases}$$

例 7.10　? SGN(5)↙

1

? SGN(0)↙

0

? SGN(-5)↙

-1

7. 求随机数

格式:RND(X)

作用:根据 X 的值来产生一个随机数。原则是:

若 RND(X) $\begin{cases}X>0,产生\ 0\sim0.999999999\ 之间的一个随机数。\\X=0,产生与前一个随机数相同的随机数。\\X<0,产生与\ X\ 相对应的一个随机数。\end{cases}$

例 7.11　产生一系列随机数。

```
10  PRINT "RND(1)="; RND (1)
20  PRINT "RND(1)="; RND (1)
30  PRINT "RND(5)="; RND (5)
40  PRINT "RND(0)="; RND (0)
50  PRINT "RND(0)="; RND (0)
60  PRINT "RND(-5)="; RND ( - 5)
70  PRINT "RND(-3)="; RND ( - 3)
80  PRINT "RND(-5)="; RND ( - 5)
90  PRINT "RND(-3)="; RND ( - 3)
100  END
```

```
]RUN
RND(1)=.270011996
RND(1)=.139756248
RND(5)=.690102028
RND(0)=.690102028
RND(0)=.690102028
RND(-5)=3.73720468E-08
RND(-3)=4.48217179E-08
RND(-5)=3.73720468E-08
RND(-3)=4.48217179E-08
```

考察以上算术运算函数，它们有如下一些共同点：

①提供的参数可以是算术表达式。

②运算的结果均是数值。

③三角函数要求参数均用弧度。

在使用这些函数时还要注意：

①正确提供实在参数。例如有些参数要求是正数值，有些参数有范围限制，如求反正切值 arctgX，要求 X 的值在 $-\frac{\pi}{2} \sim +\frac{\pi}{2}$ 之间。

②BASIC 函数同数学函数在写法上的区别。

③函数是表达式而不是语句，因为它只能进行函数运算而不能提供操作动作。它可以是语句的组成部分。

7.2 字符串处理函数

为进行字符串处理，BASIC 语言中提供了如下的字符串函数。

1. 取左子字符串

格式:LEFT $ (A $,n)

作用:取字符串 A $ 中左边的 n 个字符。

例 7.12　取字符串 A $ 中左边的 4 个字符。

```
10 A$ = "ABCDEFGH"
20 B$ =  LEFT$ (A$,4)
30  PRINT "B$=";B$
40  END

]RUN
B$=ABCD
```

2. 取右子字符串

格式:RIGHT $ (A $,n)

作用:取出字符串 A $ 中右边的 n 个字符。

例 7.13　取出 A $ 中右边的 4 个字符。

```
10 A$ = "ABCDEFGH"
20 B$ =  RIGHT$ (A$,4)
30  PRINT "B$=";B$
40  END

]RUN
B$=EFGH
```

3. 取中间子字符串

格式:MID $ (A $,n,m)

作用:取字符串 A $ 中从第 n 个字符开始的 m 个字符。

其中:n 表示开始字符位置,m 表示字符的个数。

例 7.14　取出 A $ 中间的字符。

```
10 A$ = "ABCDEFGH"
20 B$ =  MID$ (A$,3,4)
30  PRINT "B$=";B$
40  END

]RUN
B$=CDEF
```

对于 MID $ 函数,需要注意以下几点:

①如果 m 被省略,就表示取从第 n 个字符开始直到最后那个字符之间的所有字符。

②若 n 超过实际字符个数,则产生一个空串(屏幕上没有任何表示)。

③若 m 大于实际字符的个数,则有多少字符就取多少字符。

④$1\leqslant n\leqslant 255$,$1\leqslant m\leqslant 255$,若 m,n 值超界,则出现"ILLEGAL QUANTITY ERROR",表示不合法的参数出错。

4. 求字符串的长度

格式:LEN(A $)

作用:检测字符串 A $ 中有多少个字符。

例 7.15　求 A $ 的字符串长度。

```
10 A$ = "ABCDEFGH"
20 B =  LEN (A$)
30  PRINT "B=";B
40  END

]RUN
B=8
```

对于字符串处理函数必须注意：

①函数中用到的 A $ 可以是一个字符串常数或字符串变量，也可以是字符串表达式。

②n 和 m 可以是算术表达式。

③字符串长度函数得到的是一个算术值。

7.3　数据类型转换函数

BASIC 提供了将数值型转换成字符型数据以及把字符型（仅由数字开头的字符串）转换成数值型数据的数据类型转换函数。

1. 将数值型转换成字符型

格式：STR $（〈算术表达式〉）

作用：将给定的算术表达式的值由数值型转换成字符型。

例 7.16　将数值型转换成字符型。

```
? STR $ (10*10)↙
100     (这是字符 100)

10 A = 10 * 10
20 B$ =  STR$ (A)
30  PRINT "B$=";B$
40  END

]RUN
B$=100

? STR $ (3.72*10^ 15)↙
3.72E+15     (字符串)
```

数值型转换成字符型以后就可参与字符串的运算了。

2. 将字符型转换成数值型

格式:VAL $(〈串表达式〉)

作用:将由数字开头的字符串转换成数值。

例 7.17 将"123+ABC"转换成数值。

? VAL("123+ABC")↙

123 (数值)

? VAL("123456789")↙

123456789 (数值)

? VAL("A123+B456")↙

0

对于 VAL 函数,并非所有字符串都能转换成数值。对于全部由数字符号组成的字符串,VAL 函数可以将其完整地转换成数值。对于仅由数字开头的字符串,VAL 只能将开头连续的数字符号转换成数值。对于非数字符号开头的字符串,VAL 只能转换成数值 0。

3. 求数值对应的 ASCII 码字符

格式:CHR $(〈算术表达式〉)

作用:给出算术表达式值所对应的 ASCII 码字符。

例 7.18 求 0～255 之间的 ASCII 码字符。

```
10  FOR I=0 TO 255
20  PRINT CHR $ (I);" ";
25  IF INT(I/10) * 10=I THEN PRINT
30  NEXT I
40  END
RUN↙
```

屏幕上显示出所有字符(这里省略了)。第 25 句的作用是使每 10 个符号占一行,便于浏览。

注意：

①算术表达式的值必须在0～255之间。

②若算术表达式的值为实数，机器自动转换成整数后再进行函数运算。

③用该函数可以得到键盘上没有的许多符号。

4. 求字符的ASCII码

格式：ASC$(〈字符串表达式〉)

作用：返回字符串中第一个字符的ASCII码值。

例7.19 求A的ASCII码值。

？ASC("ABCD")↙

65　　　　(字母A的ASCII码值是65)

？ASC("WXYZ")↙

87　　　　(字母W的ASCII码值是87)

注意：该函数每次只能返回字符串表达式中最左边那个字符的ASCII码值，而不能给出串中全部字符的ASCII码值。

7.4 控制输出格式的函数和命令

前面我们介绍和使用了PRINT语句，其一般格式应该是：PRINT〔打印格式〕〔表达式〕，其中的打印格式是表达式之后紧跟分号或逗号，这种情况我们已经介绍了。下面再介绍几个与之有关的函数。

7.4.1 格式及其作用

1. 空格函数

格式：SPC(〈算术表达式〉)

作用：以上一次输出的最后那个字符为基准，跳过若干个空

格。算术表达式的值就是应该跳过的空格的个数。

例 7.20　在两个输出数据间留出适当空格。

```
10 PRINT 125;SPC(3);6789
20 END
RUN↙
125         6789
```

在数字 125 的 5 之后留出 3 个空格。即是说 125 和 6789 之间有 3 个空格。

注意：

①SPC 函数只能放在 PRINT 之后用于跳过若干空格。

②算术表达式的值指出了空格的个数，其值必须在 0～255 之间。

2. 将光标移到指定的行上

格式：VTAB(〈算术表达式〉)

作用：将光标移到算术表达式的值所指定的行上。

例 7.21　将光标移到第 3 行上。

```
10 PRINT 125;
20 VTAB(3)
30 END
RUN
125
```

输出数值 125 以后，光标移到屏幕的第三行上。

注意：

①VTAB 命令只能将光标上下移动而不能左右移动。如果光标当前所处的行号比 VTAB 中给出的行号小，那么光标向下移。如果当前行号比 VTAB 中指定的行号大，那么光标上移。

②在西文状态下，算术表达式的值只能在 1～24 之间(屏幕上所能显示字符的行号范围)。中文状态下只能在 1～10 之间。

3. 将光标移动若干列

格式:HTAB(〈算术表达式〉)

作用:将光标移动指定的列数。数据从该列开始显示或打印。

例 7.22　将光标右移 10 列。

```
10 PRINT 125;
20 HTAB (10)
30 PRINT 789
40 END
RUN
125             789
```

数值 789 从第 10 列开始显示,这就是 HTAB(10)的作用。

注意:

①算术表达式的值给出了左右移动的列数。

②在西文状态下,1～40 为当前行(即光标所在的行),41～80 列为下一行;在中文状态下,1～34 列为当前行,35～68 列为下一行。

③算术表达式的值只能在 0～255 之间。

4. 将光标定位在指定的列上

格式:TAB(〈算术表达式〉)

作用:以当前行上的第一列为基准,将光标定位在指定的列上。

例 7.23　在指定位置上开始打印数据。

```
10 PRINT 125;TAB(10);6789;
20 PRINT TAB(20);"AAAAA"
30 END
RUN↙
125          6789               AAAAA
```

注意：

①光标只能向右移而不能回退。即是说，如果先用了 TAB (20)，就不能再在同一行上用 TAB(15)。

②TAB 只能在 PRINT 中使用。

③算术表达式的值只能在 0～255 之间。

5. 求光标所处的列位置

格式：POS(X)

作用：返回光标所在的列号。如：

? POS(5)↙

3　　　　(表示光标现在正处于第 3 列上)

注意：这里的 X 是无所谓的，实际上不起任何控制作用。因而该函数可以写成 POS(0)。写其它值效果也一样。

7.4.2 打印格式的控制

在前面介绍的格式函数中，用得最多的是 TAB 函数。它可以使我们方便地把需要打印的字符安排在适当的列位置上。我们用几个例子来说明如何使用 TAB 函数。

例 7.24 打印如图 7.1 所示的图形。

```
10  FOR I = 1 TO 8
20  FOR J = 1 TO I
30  PRINT  TAB( 15 + J);"A";
40  NEXT J
50  PRINT
60  NEXT I
70  END

]RUN
```

```
A
AA
AAA
AAAA
AAAAA
AAAAAA
AAAAAAA
AAAAAAAA
```

图 7.1

运行结果如图 7.1 所示。

例 7.25　打印如图 7.2 所示图形。

```
10  FOR I = 1 TO 8
20  FOR J = 1 TO 10
30  PRINT  TAB( 5 + I + J);"A";
40  NEXT J
50  PRINT
60  NEXT I
70  END

]RUN
```

```
AAAAAAAAAA
 AAAAAAAAAA
  AAAAAAAAAA
   AAAAAAAAAA
    AAAAAAAAAA
     AAAAAAAAAA
      AAAAAAAAAA
       AAAAAAAAAA
```

图 7.2

运行结果如图 7.2 所示。

上面几个例子中用到了 TAB 函数，其作用均是控制字符打印的位置。要使字符打印在所希望的列位置上，关键是设法使 TAB(〈算术表达式〉)中算术表达式的值能够恰到好处。只要找出了打印字符的排列规律，算术表达式就不难构造出来了。

7.5　自定义函数

以上介绍的这些函数是由系统提供给用户直接使用的，通常称为标准函数。这些函数功能专一，调用简便。为使用户能够象使用标准函数那样方便，BASIC 又提供了用户定义自己所需函数的能力，这就是所谓的自定义函数。

格式：DEF FN〈自定义函数名〉(〈形式参数〉)

　　　=〈算术表达式〉

作用：允许用户定义自己所需的函数。

其中:DEF FN 是自定义函数的标志。自定义函数名的构成法同变量名相同。形式参数是一个实型算术变量,算术表达式是需要计算的式子。

我们用例子来说明如何定义和使用自定义函数。

例 7.26　求下式的值。

$$W=\frac{b-a}{4}\left(\frac{\sqrt{1+a^2}}{2}+2\sqrt{1+(a+h)^2}+\frac{\sqrt{1+(a+2h)^2}}{2}\right)$$

其中:$a=5, b=8, h=\frac{b-a}{2}$

我们用自定义函数来解决该问题。

```
10  DEF  FN F(X) =  SQR (1 + X * X)
20 A = 5:B = 8:H = (B - A) / 2
30 W = (B - A) / 4 * ( FN F(A) / 2 + 2 *  FN F(A + H) +  FN
F(A + 2 * H) / 2)
40  PRINT "W=";W
50  END

]RUN
W=14.8001888
```

说明:

10 句定义了一个自定义函数 F(X),它用于计算 $F(X)=\sqrt{1+X^2}$式子的值。

30 句中调用了 10 句中定义的函数 F(X)。函数 F(X)如同标准函数的使用一样方便。

使用自定义函数时必须注意:

①自定义函数必须是数值型函数,不允许是字符串型。

②自定义函数必须先定义后使用。

③自定义函数中至多只能有一个形式参数，该形式参数在算术表达式中可以出现多次，也可以一次都不出现。

④同一程序中可以对同一自定义函数进行重新定义。

例 7.27　对自定义函数的重新定义。

```
10  DEF  FN F(X) =  SQR (1 + X * X)
20 W1 =  FN F(5) +  FN F(8)
30  DEF  FN F(Y) = ( LOG (Y) +  EXP (Y)) ^ 2
40 W2 =  FN F(3)
50  PRINT "W1=";W1;"  ";"W2=";W2
60  END

]RUN
W1=13.1612773  W2=448.768179
```

在上述程序中，第 10 句定义了自定义函数 F(X)，但在第 30 句中又对自定义函数进行了重新定义，这是允许的。

⑤在调用自定义函数时，必须写成 FN〈变量名〉(〈实在参数〉)的形式。如上例中的 FN F(5)，FNP(0)等。自定义函数是一种算术表达式，因而，凡是算术表达式能够出现的地方，自定义函数均可出现。

问题与思考

1. 自定义函数与标准函数有何区别？

习题七

1. 用自定义函数编写程序。

$$M=\frac{1+\sin\alpha+\sin^2\alpha}{1+\cos\alpha}\left(\sqrt{\frac{1+\sin\beta+\sin^2\beta}{1+\cos\beta}}-\frac{1+\sin\gamma+\sin^2\gamma}{1+\cos\gamma}\right)$$

其中：$\alpha=30°\quad\beta=38°\quad\gamma=58°$

2. 编写打印九九表的程序。排列形式为：

1 * 1=1

2 * 1=2　2 * 2=4

3 * 1=3　3 * 2=6　3 * 3=9

4 * 1=4　4 * 2=8　4 * 3=12　4 * 4=16

5 * 1=5　5 * 2=10　5 * 3=15　5 * 4=20　5 * 5=25

……

3. 编写打印下列学生成绩统计表的程序。

学生成绩统计表

学号	姓名	语文	政治	数学	物理	外语	化学	总分
613	张珊	85	78	82	73	77	70	…
624	李实	92	91	88	85	85	80	…
⋮	…							

第八章　子程序

重复执行某段程序，在大多数情况下利用循环语句来实现是方便的。但在某些情况下用子程序技术却更为简便。

8.1　转子语句和返回语句

在实际的程序设计中，我们经常都会碰到这样的问题：一个大型问题中包含了若干相对独立的小问题。对于这种情况，程序设计的一般经验是将大问题用一个总控程序来实现，称为主程序。每一个相对独立的小问题各自单独对应一个程序，称为子程序。当主程序中需要解决某个小问题时，就调用它所对应的子程序。通常，主程序向子程序提供数据，子程序向主程序返回运算结果。

8.1.1　语句格式及其作用

格式 1：GOSUB 〈行号〉

作用：无条件转向〈行号〉所代表的子程序去执行，当遇到子程序中的返回语句时再把控制转到该转子语句的后继语句。

其中：GOSUB 是转子语句的标志。行号是子程序的入口语句行号。

格式 2：ON 〈n〉 GOSUB 〈行号 1〉〔,〈行号 2〉…〕

作用：根据 n 的值转到相应行号所代表的子程序去执行。

其中：ON n 是转子条件，n 是一个算术表达式且其值是整

数(0⩽n⩽255)。如果 n 为实数,机器将自动转换为整数。如果 n=0 或 n 大于子程序的最大行号,那么就执行转子语句的后继语句。n 的值应该同指出的子程序入口语句行号的顺序号相对应。n=1,转到行号 1,n=2,转到行号 2,其余类推。

格式:RETURN

作用:从子程序返回到相应转子语句的后继语句去执行。

RETURN 这几个字母是返回语句的标志。它表示依次敲入 R、E、T、U、R、N 这几个字母,而不是代表回车键。一个子程序中至少有一个返回语句。

如何利用转子语句和返回语句来编制程序呢?我们看下面的例子。

例 8.1 用主程序和子程序的形式来编写程序。

$$W=\frac{b-a}{4}\left(\frac{\sqrt{1+a^2}-\sqrt[3]{1+a^2}}{2}+4\frac{\sqrt{1+(a+h)^2}-\sqrt[3]{1+(a+h)^2}}{2}+\frac{\sqrt{1+(a+2h)^2}-\sqrt[3]{1+(a+2h)^2}}{2}\right)$$

其中:$a=5, b=8, h=\frac{b-a}{2}$,在上式中,$\frac{\sqrt{1+x^2}-\sqrt[3]{1+x^2}}{2}$形式的式子出现了三次,我们就把该式作为子程序。

写出主程序如下:

```
10 A = 5:B = 8:H = (B - A) / 2
20 X = A
30  GOSUB 1000
40 F1 = F
50 X = A + H
60  GOSUB 1000
70 F2 = F
```

```
80 X = A + 2 * H
90  GOSUB 1000
100 F3 = F
110 W = (B - A) / 4 * (F1 + 4 * F2 + F3)
120  PRINT "W=";W
130  END
```

再根据要求写出子程序：

```
1000 S = X * X
1010 F = ( SQR (1 + S) - (1 + S) ^ (1 / 3)) / 2
1020  RETURN

]RUN
W=6.9162189
```

说明：

10～130 句是主程序段。它的作用是向子程序提供数据，保存子程序中返回的运算结果。

20,50,80 句均是向子程序提供数据。

30,60,90 句均是调用子程序。子程序入口语句的行号是 1000。

40,70,100 句是保存子程序中返回的计算结果。

110 句是利用保存的结果计算 W 的值。

130 句是主程序的结束标志。在带有子程序的程序设计中，主程序一定要有 END 语句(命令)。

1000～1020 句是子程序。显然，这里的子程序相当简单。子程序中至少有一个 RETURN 语句。

主程序和子程序的执行情况如图 8.1 所示。



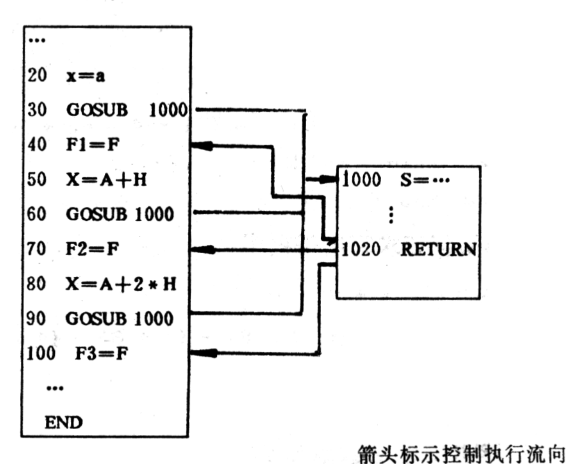


图 8.1　主-子程序执行流程示意图

20 句向子程序中的变量 X 提供数据，30 句转到行号为 1000 的子程序入口语句去执行，当执行到子程序的 RETURN 语句时，控制又返回到主程序中相应转子语句(30 句)的后继语句(40 句)去执行。第 40 句把子程序中计算的结果 F 保存到中间变量 F1 中。第 50 句向 X 提供数据 A+H。第 60 句又转到子程序去执行。重复以上过程，直到调用完毕。

通过分析以上主-子程序的执行情况，我们可以得出主程序和子程序之间调用的一般方式：主程序向子程序提供数据——转向子程序——执行子程序——返回主程序——子程序向主程序提交运行结果——再执行主程序。

例 8.2　求 $ax^2+bx+c=0$ 的根。其中 a，b，c 可以取不同的几组数据。为便于比较，请将方程的根列成表格形式，如下所示：

A	B	C	X_1	X_2	备注
3	4	5	…	…	…
8	20	10	…	…	…
……					

备注项用于说明方程得到的是实根还是虚根(如果是虚根，那么表中 X_1 是实部，X_2 代表虚部)。

我们把提供数据(a,b,c 的值)作为主程序，把求解方程的根作为子程序 1(入口是 1000)。把格式打印作为子程序 2(入口是 5000)。请看下面的程序：

```
10  PRINT "————————————————————————————————————"
15  PRINT "|"; TAB( 3);"A"; TAB( 6);"B"; TAB( 9);"C";
20  PRINT  TAB( 14);"X1"; TAB( 22);"X2"; TAB( 29);
    "备注"; TA  B( 36);"|"
25  PRINT "————————————————————————————————————"
30  FOR I = 1 TO 5
40  READ A,B,C
50  GOSUB 1000
60  GOSUB 5000
70  NEXT I
80  PRINT "程序结束"
90  DATA  3,4,5,8,20,10,17,19,6,24,56,43,13,21,8
100  END
1000 D = B * B - 4 * A * C
1100  IF D > 0 THEN 1700
1200  IF D = 0 THEN 2100
1300 X1 =  - B / (2 * A)
1305 X1 =  INT (X1 * 10000) / 10000
1400 X2 =  SQR ( - D) / (2 * A)
```

```
1405 X2 =  INT (X2 * 10000) / 10000
1500 G$ = "共轭虚根"
1600  RETURN
1700 X1 = ( - B +  SQR (D)) / (2 * A)
1705 X1 =  INT (X1 * 10000) / 10000
1800 X2 = ( - B -  SQR (D)) / (2 * A)
1805 X2 =  INT (X2 * 10000) / 10000
1900 G$ = "不等实根"
2000  RETURN
2100 X1 =  - D / (2 * A)
2105 X1 =  INT (X1 * 10000) / 10000
2200 X2 = X1
2300 G$ = "相等实根"
2400  RETURN
5000  PRINT "!"; TAB( 3);A; TAB( 6);B; TAB( 9);C;
5100  PRINT  TAB( 12);X1; TAB( 20);X2; TAB( 28);G$;
      TAB( 36) ;"|"
5200  PRINT "________________________________________"
5300  RETURN

]RUN
_____________________________________
! A  B  C    X1      X2      备注   !
_____________________________________
! 3  4  5  -.6667  1.1055   共轭虚根 !
_____________________________________
! 8  20 10 -.691   -1.8091  不等实根 !
_____________________________________
! 17 19 6  -.5589  .2016    共轭虚根 !
_____________________________________
! 24 56 43 -1.1667 .6561    共轭虚根 !
_____________________________________
! 13 21 8  -.6154  -1       不等实根 !
_____________________________________
```

10～100 句是主程序，主要用于提供 a，b，c 的值。10～25 句用于打印表格的栏目。

1000～2400 句是子程序 1，主要用于求解方程的根。由于我们需要把方程的系数和根用表格的形式列出，因而对 X_1，X_2 的值我们仅保留 4 位小数。采取的办法就是将 X_1，X_2 扩大 10000 倍取整，然后再缩小 10000。这是由子程序 1 中的 1305，1405，1705，1805 和 2105 等语句来实现的。当然，我们也可以把对 X_1，X_2 保留四位小数的操作放到子程序 2 中去统一处理，这样子程序 1 可以更加精炼（请读者自己改进）。子程序 1 中用了三个 RETURN 语句，这是允许的。

5000～5300 句是子程序 2，主要用于打印数据和表格的横线。为了简单，横线可以用虚线。为使表格美观可以使用制表符。方法是敲 F2 键进入拼音方式后再敲\（斜线），屏幕显示出制表符，这时可以象汉字那样取而用之。

8.1.2 编写和调用子程序时注意

1. 主程序和子程序的语句行号不能交叉。主程序必须用 END 作为结束标志，子程序必须有 RETURN 返回主程序。参见图 8.2 和图 8.3。

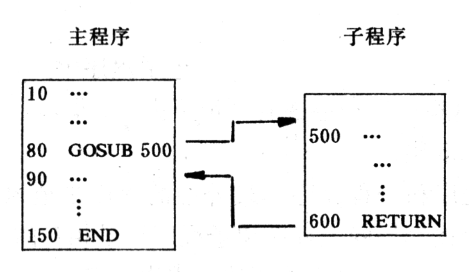


图 8.2 正确的语句行号

在图 8.3 所示的主程序和子程序中，主程序的语句编号(200)和子程序的语句行号(200)相交了，这是不行的。

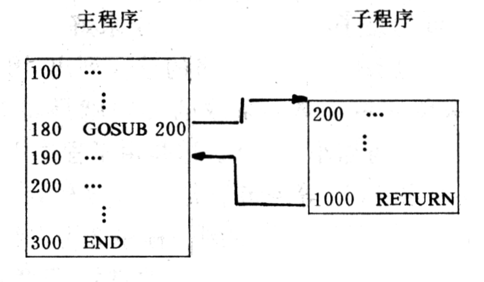


图 8.3 不正确的语句行号

2. 主程序要为子程序提供适当的数据。子程序运算的结果可以返回给主程序。

3. 子程序中至少应有一个返回语句 RETURN。执行子程序遇到 RETURN 时就返回到调用语句的后继语句去执行。如果子程序中没有返回语句，子程序执行结束后就无法回到主程序中相应的后继语句。

4. 调用子程序只能用转子语句 GOSUB，而不能用 GOTO 或 IF—THEN 语句。因为一般调用子程序的目的是希望执行完子程序以后再返回到主程序，只有 GOSUB 配合 RETURN 才能达到这个目的。

5. 子程序可以嵌套调用。即一个子程序中又可以调用另外的子程序。这样一层一层嵌套调用下去。中华学习机 CEC- I 最多允许嵌套 25 层。

6. 主程序和子程序除了最外层的程序是地地道道的主程序以外，其余嵌套层的主程序和子程序均是相对的。即是说，某一个主程序可以是另一个程序的子程序。多重子程序的嵌套和执行情况如图 8.4 所示。

当然，子程序 1 除了调用子程序 2 以外，还可以调用其它子程序。但一般不得递归调用。所谓递归调用是指这种情况：主程序调用子程序 1，子程序 1 调用子程序 2，子程序 2 又调用子程序 3，而子程序 3 又反过来调用子程序 1 或主程序，或者是调用自身。如图 8.5 所示。

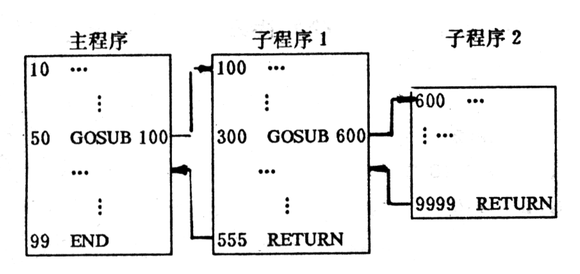


图 8.4　多重子程序调用示意图

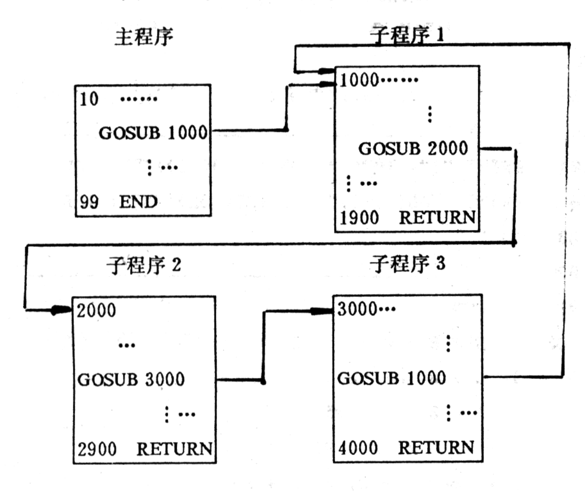


图 8.5　递归调用示意图

只要不造成递归调用，子程序的调用还是比较自由的。任何子程序都不能调用最外层的主程序。我们来看一个多重调用的

例子。

例 8.3 子程序的多重调用。仍然用求一元二次方程根为例(若为虚根,那么 X_1 代表实部,X_2 代表虚部)。

我们把打印表头和栏目作为子程序 1(入口是 1000),求解方程的根作为子程序 2(入口是 2000),打印表格作为子程序 3(入口是 3000)。提供数据的部分作为主程序。

```
10  GOSUB 1000
20  FOR I = 1 TO 5
30  READ A,B,C
40  GOSUB 2000
50  NEXT I
60  PRINT "程序结束"
70  DATA  3,4,5,8,20,10,17,19,6,24,56,43,13,21,8
80  END
1000  PRINT  TAB( 10);"求解方程根一览表"
1005  PRINT "————————————————————————————————————"
1010  PRINT "|"; TAB( 3);"A"; TAB( 6);"B"; TAB( 9);"C";
1015  PRINT  TAB( 14);"X1"; TAB( 22);"X2"; TAB( 29);"备注";
TAB( 36);"|"
1020  PRINT "————————————————————————————————————"
1025  RETURN
2000 D = B * B - 4 * A * C
2005  IF D > 0 THEN 2035
2010  IF D = 0 THEN 2055
2015 X1 =  - B / (2 * A)
2020 X2 =  SQR ( - D) / (2 * A)
```

```
2025 G$ = "共轭虚根"
2028  GOSUB 3000
2030  RETURN
2035 X1 = ( - B +  SQR (D)) / (2 * A)
2040 X2 = ( - B -  SQR (D)) / (2 * A)
2045 G$ = "不等实根"
2048  GOSUB 3000
2050  RETURN
2055 X1 =  - B / (2 * A)
2060 X2 = X1
2065 G$ = "相同实根"
2068  GOSUB 3000
2070  RETURN
3000 X1 =  INT (X1 * 10000) / 10000
3005 X2 =  INT (X2 * 10000) / 10000
3010  PRINT "!"; TAB( 3);A; TAB( 6);B; TAB( 9);C;
3015  PRINT  TAB( 12);X1; TAB( 20);X2; TAB( 28);G$; TAB( 35)
;"!"
3020  PRINT "————————————————————————————————————"
3025  RETURN
```

运行结果省略了。程序的执行情况如图 8.6 所示。

ON 〈n〉 GOSUB 〈行号〉…这个形式的转子语句用法类似 ON 〈n〉 GOTO 〈行号〉…条件转语句。关键仍然是如何构造 n。在介绍 ON 〈n〉 GOTO 〈行号〉时已经有一些例子,请读者自己改用 ON 〈n〉 GOSUB 后上机一试。

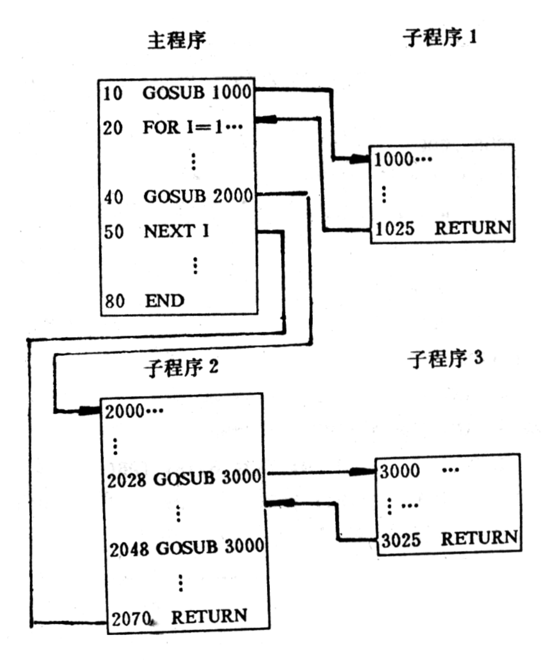


图 8.6 主子程序调用示意图

8.2 子程序应用实例

如果我们乐意的话，任何问题都可以用主一子程序的形式来编写。

例 8.4 求 $W=\frac{N!}{(A-B)!A!}$ 的值，其中 A，B，H 为已知数。

```
10  INPUT A,B,N
20 K = N
30  GOSUB 1000
40 W1 = M
```

```
50 K = A
60  GOSUB 1000
70 W2 = M
80 K = A - B
90  GOSUB 1000
100 W3 = M
110 W = W1 / (W2 * W3)
120  PRINT "W=";W
130  END
1000 M = 1
1005  FOR I = K TO 1 STEP  - 1
1010 M = M * I
1015  NEXT I
1020  RETURN

]RUN
?10,5,15
W=3003
```

在上机试用该程序时，数据不能太大，否则会发生溢出。

例 8.5　求多边形的面积。

计算多边形的面积问题可以转换成求各个三角形的面积之和的问题。

求三角形面积的公式为：

$$S=\sqrt{P(P-a)(P-b)(P-c)},$$

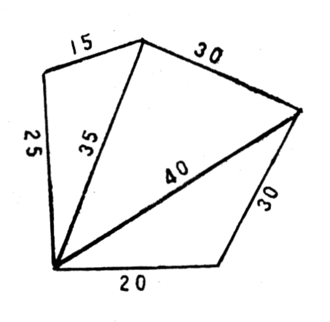


图 8.7 例 8.5 图

$$P=\frac{a+b+C}{2},$$

其中 a,b,c 为三角形的三边长度。

程序如下：

```
10 W = 0
20  FOR I = 1 TO 3
30  READ A,B,C
40  GOSUB 1000
50 W = W + S
60  NEXT I
70  PRINT "多边形的面积=";W
80  DATA  20,30,40,40,30,35,35,25,15
90  END
1000 P = (A + B + C) / 2
1010 S =  SQR (P * (P - A) * (P - B) * (P - C))
1020  RETURN

]RUN
多边形的面积=961.182579
```

问题与思考

1. 为什么不能用 GOTO 转向子程序？如果一定要用 GOTO 转到子程序，那么可能会出现什么情况？

2. 为什么子程序中必须有至少一个 RETURN 语句？在子

程序中出现 RETURN 的地方用 GOTO 转回主程序可以吗？哪些情况下可以？哪种情况下不可以？

习题八

1. 指出下列程序的运行结果。

```
10   PRINT "主程序"
20   PRINT "AAA1"
30   GOSUB 1000
40   PRINT "AAA2"
50   GOSUB 2000
60   GOSUB 3000
70   PRINT "BBB1"
80   END
1000   PRINT "子程序 1"
1100   PRINT "CCC1"
1200   GOSUB 2000
1300   RETURN
2000   PRINT "子程序 2"
2100   PRINT "CCC2"
2200   RETURN
3000   PRINT "子程序 3"
3100   PRINT "DDD1"
3200   GOTO 1000
3300   RETURN
RUN↙
```

2. 指出程序的书写错误。

①

```
10   PRINT "主程序"
20   GOSUB 1000
```

```
    30  PRINT "AAA"
    40  END
  1000  PRINT "BBB"
  1100  GOSUB 2000
  1200  PRINT "BBB"
  1300  RETURN
  2000  PRINT "CCC"
  2100  GOSUB 1000
  2200  PRINT "DDD"
  2300  RETURN
```

②

```
    10  PRINT "主程序"
    20  GOSUB 1000
    30  PRINT "AAA"
    40  END
  1000  PRINT "BBB"
  1100  GOSUB 2000
  1200  PRINT "BBB"
  1300  RETURN
  2000  PRINT "CCC"
  2100  GOSUB 10
  2200  PRINT "DDD"
  2300  RETURN
```

3. 用子程序编写下式程序。

①$W=\frac{1-a}{1+a^2+a^5}(\frac{1-b}{1+b^2+b^5}+\frac{1-c}{1+c^2+c^5})$

$\frac{1-(x+y)}{1-(x+y)^2+(x+y)^5}$

②设 x,y 为已知数值,求

$y=\frac{f(x)+f(y^2)}{f(x+3)+y^3}f(x+y),f(u)=u-\frac{u^3}{6}+\frac{u^5}{120}$

4. 在发放工资时，财会人员不但要计算出本部门的所有职工的总工资，而且还必须把每个人的工资进行分解，以便知道需要多少张拾元卷、伍元卷、壹元卷、伍角票、贰角票、壹角票、伍分票、贰分票和壹分票。请编写一个程序，接收每个人的工资额后自动进行分解，并累计本部门发放工资时各种面额的钞票各自需要多少张。要求要用到子程序。

第九章　文件管理系统

在实际应用中，我们常常希望将一些数据、报表或程序保存起来，以备随时使用。要保存这些数据、报表或程序，就要用到中华学习机 CEC-Ⅰ的主要外部设备——磁盘驱动器和软磁盘（当然也可以用录音机和磁带）。除了这些硬设备以外，还要用到中华学习机上的文件管理系统。软盘驱动器的用法是简单和直观的，前面已经作了介绍。本章主要介绍如何使用文件管理系统在软盘上建立用户的程序文件和数据文件。

9.1　文件的基本概念

中华学习机 CEC-Ⅰ把用来存放数据信息的文件都称为文本文件，因为文件是由文本字符构成的。这些文本字符可以是：运算的数据，计算的结果，待输出的报表以及 BASIC 程序。文本文件是通过在 BASIC 程序中调用操作系统的命令来建立和检索的。它可以由一个程序来建立，用另一个程序来调用。

现在先介绍几个与文件有关的概念。

9.1.1　数据、字段、记录、文件

1. 数据

在介绍 BASIC 语句中已经涉及到了数据这个概念。但那里对数据的说法是比较单纯的，其主要含义是指通常用到的数值常数或字符串常数。在文件管理中，我们把数据一词的含义拓广

了，它不仅可以指数值常数、字符串变量的值，也可以指一个报表，一段程序。总之，我们要用到的任何符号均可以看成是数据。在文件管理中，数据可以用来表示我们所关心的任何信息。数据和信息是两个既有联系又有区别的概念。信息有一定的含义和意思。在计算机中，信息总是以数据的形式表现出来。数据不一定有什么意思，它是载荷信息的一些符号。为了简单，有时不加区别地使用数据和信息这两个术语。总起来说，凡是计算机要处理的东西，我们都可以认为是数据。数据是一个含义非常广泛的术语。

2. 字段

我们先来看表 9.1。

表 9.1　学生情况统计表

姓名	学号	性别	年龄	成绩
张　山	631	男	15	95
李丽丽	632	女	13	88.5
王大力	634	男	14	74.7

在表格中列出了姓名、学号、性别、年龄和成绩共 5 个栏目。每一个栏目在文件管理中就称为一个字段(Field)。每一个字段下面都对应了一些数据。如姓名字段下面有张山、李丽丽和王大力三个数据。这三个数据中的每一个数据都叫姓名字段的值(实际上就是表格的每一列)。一个表格有多少栏目(列)，就有多少不同的字段。每一个字段可以取若干个值。总之，我们可以把表格中的栏目同字段等同起来，或者干脆就称为栏目。

3. 记录

上面的表格共有 5 个字段，这 5 个字段合起来就称为一个记录。若干个字段组成一个记录。张山　631　男　15　95 这是

一个记录。而李丽丽　632　女　13　88.5 则是另一个记录。实际上，表格中的一个横行就表示一个记录。表格中有几个横行就表示有几个记录。一个记录应当包含几个字段，这是由实际应用来确定的，当然可以很多，也可以比较少。但中华学习机 CEC-Ⅰ要求一个横行（一个记录）的所有字符加起来不能超过 256 个。如果一个实际问题中一个记录可能超过 256 个字符，就只好分成多个表格来处理了。

从上面给出的表格我们可以看到：字段是按列来划分的，而记录则是按行来划分的。所以，以后我们又可以把字段与列、记录与行等同使用。

4. 文件

若干个记录组成一个文件。前面的学生情况统计表，它有三个记录，可以组成一个学生文件。一个文件可以同整个表格联系起来。一个文件可以代表一个表格，一个表格也可以表示一个文件。表格和文件可以等同使用。在用文件管理系统来描述文件（表格）时，首先必须给文件取一个名字，称为文件名。文件名是用字母开头的字符序列（不能含有逗号），最多可以有 30 个字符，也可以直接用汉字作文件名。但必须注意，一个汉字相当于两个英文字符的长度。因而汉字文件名最多包含 15 个汉字。

如果一个文件中有多个记录，那么，为了使记录有一个先后次序，因而必须给每个记录编一个顺序号，称为记录号。记录总是从 0 开始编号。例如学生情况统计的记录可按表 9.2 编号。

如果还有其它记录，可以依次编下去。这种编号可以由机器自动进行（如果由机器自动进行，那么就按输入记录的先后顺序依次编号）。

顺便介绍一下记录指针的概念。所谓记录指针，就是为了能够自由的存取文件记录而设置的一个计数器，它能够指示记录

在文件中的位置。我们要存取某一个记录，首先就必须使记录指针指向该记录。根据需要，可以使指针在各个记录位置上来回拨动。一个文件打开以后，指针总是指向文件的当前记录（通常是第一个记录即 0 号记录），随着对文件的操作，指针位置可以不断变化。

表 9.2　学生情况统计的记录

0	张　山	631	男	15	95
1	李丽丽	632	女	13	88.5
2	王大力	634	男	14	74.7
…	…				

归纳起来，我们可以得出以下结论：

①一个文件由若干个记录组成，文件名是这些记录的总代表。特殊情况下，一个文件也可以只有一个记录或没有任何记录（只有一个文件名）。

②一个记录由若干个字段组成。一个记录可以包含一个或多个字段。

③一个字段由若干个字符组成。一个字段中可以包含一个或多个字符。

④同类事物的数据可以组成一个文件。例如：所有学生的成绩档案组成一个学生文件。所有职工的工资情况组成一个工资文件。所有商品信息组成一个商品文件。最好不要将不同类型的数据组织在同一个文件中。如学生的学习成绩和库存物资，这是两码事，可以分别组织成两个文件。

9.1.2　文件的种类

简单地说，中华学习机上我们经常用到的文件有三种

专门用于保存 BASIC 程序的文件，称为程序文件。另两种是用于保存数据信息的文件，称为数据文件，又称文本文件。

1. 程序文件

所谓程序文件就是用 BASIC 语句或 DOS 命令编写的程序所组成的文件。在输入、调试、运行 BASIC 程序(特别是很长的程序)时，为了防止程序遭到意外的破坏(如：突然断电，机器故障，人为操作失误等)，在调试、修改和运行之前，最好先将程序保存在软盘上，当内存被破坏时，再从软盘上把程序调到内存，重新开始调试、修改和运行。这样，不至于使我们辛辛苦苦输入到机内的程序毁于一旦。为此就要用到程序文件。

2. 顺序文件

顺序文件由字段组成，每个字段值之后用回车(CR)字符作为终结标志。当然，我们也可以用记录值来组成顺序文件，每个记录值之后同样用回车字符作为结束标志(这种情况相当于把整个记录值作为一个字段值)。顺序文件的数据在软盘上存放时是依次连续存放的，即一个字段(记录)接一个字段，各个字段值之间没有空格(间隙)。顺序文件的这种顺序存储方式可以看成是把所有数据拉成一条线而依次存放。字段和文件的长度均不受限制(当然要受到软盘容量的限制)。

顺序文件的特点是：建立时先送入的数据放在前面，后送入的数据排列在后面。检索时也是一样，需要检索后面的数据时，必须先检索前面的数据(从第 0 个记录起)，不管前面的数据是否需要均必须如此。例如：我们要查找第 100 个数据，必须先找出前面的 99 个数据。这如同日常排队买东西一样，先来先买。这种处理方法，用计算机的术语来说叫做“先进先出”。

顺序文件又称串行文件，其优点是：建立方式比较简单，节省存储空间，在需要顺序依次存取的应用情况下比较快速和方

便。最大的缺点是检索时必须论资排辈，依次而行，从计算机的处理这点来说，比较浪费时间而又不太方便。

3. 随机文件

随机文件又称直接文件。它由记录组成，每个记录中又可以包含若干字段。在随机文件中，各个记录的长度是固定的，一个记录的最大长度是 32767 个字符。

随机文件的特点是：建立文件时记录不一定依次连续存放，检索时可以按任意顺序检索任何记录(当然，也可以顺序检索)。文件以记录为单位进行读写操作。所谓文件的读或写是什么意思呢？写操作是指把计算机内存的数据传送到软盘的文件中(内存⇒外存)保存起来，而读操作则刚好相反，它是把外存(软盘)上文件中保存的数据传送到内存(外存⇒内存)。建立文件时传送数据必须用写(Write)操作(内存⇒外存)，检索文件时传送数据必须用读(Read)操作(外存⇒内存)。

9.2 程序文件的建立和载入

如何把 BASIC 程序用文件的形式保存在软盘上？又如何把软盘上的 BASIC 程序调入内存？

1. 建立 BASIC 程序文件的命令

格式：SAVE 〈文件名〉

作用：把内存中现有的 BASIC 程序用给定的文件名保存在软盘上。

2. 载入命令

所谓载入就是把磁盘上的文件调入内存。

格式：LOAD 〈文件名〉

作用：将先前保存在磁盘上的程序文件调入内存。SAVE 和

LOAD 命令的应用实例请参见 4.7.2 节。

需要注意的是:SAVE 和 LOAD 命令只能用于建立和载入 BASIC 程序文件。顺序文件、随机文件是不能用 SAVE 和 LOAD 命令来建立和载入的。

3. 删除文件

格式:DELETE 〈文件名〉

作用:删除指定的磁盘文件。

如果一个文件确实不再需用了,可以用删除命令把它删去,从而释放出磁盘空间供其它文件使用。需要特别强调的是:一旦用该命令把文件删去,该文件就从磁盘目录中消失了,而且不能再恢复,除非另外重新建立。所以,使用 DELETE 命令要格外谨慎。DELETE 命令可用于删除磁盘上的任何文件(包括程序文件、顺序文件、随机文件以及系统文件)。

9.3 顺序文件的建立和检索

9.3.1 顺序文件的建立方式

1. 文件的建立或打开命令

格式:OPEN 〈文件名〉 [,Ss][,Dd][,Vv]

作用:如果文件名在磁盘目录中还没有,就表示新建一个顺序文件,机器自动将文件名列入磁盘目录中。如果文件名已经存在,则表示打开指定的文件。

OPEN 命令实际上有两种可能的作用:一是建立新文件;二是打开一个旧文件。关键是文件名在磁盘目录中是否存在。如果我们取的文件名凑巧和磁盘上其它文件同名,那么仍然是作为打开文件来使用。

其中:文件名是需要建立或打开的文件的名字,它由字母开头,至多 30 个字符组成,文件名中不能含有逗号字符。

Ss——表示软盘驱动器所在的槽口号码,s 的值可以是 1～7 之间的整数(但不能是 3,3 留作汉字系统用)。中华学习机 CEC-Ⅰ上的 s=6。

Dd——软盘驱动器在盘驱接口卡上的设备号。d 值可以是 1 或者 2。中华学习机上的 d=1。

Vv——软盘片的卷号。一张软盘有唯一的卷号,这个卷号是软盘初始化时由 INIT 命令确定的。写上卷号的目的是防止软盘拿错。v 可以是 1～254 之间的一个整数。v 的值不能乱写,一定要根据初始化时确定的卷号,如果我们可以保证软盘不会拿错,Vv 项完全可以省去不写。

在实际使用 OPEN 命令时,必须嵌入到 BASIC 的 PRINT 语句中去。

例 9.1 建立一个顺序文件,文件名就是“学生”。

```
10 D$ =CHR$(4)
20 PRINT D$;"OPEN 学生,S6,D1,V254"
…
```

这就表示创建了一个“学生”顺序文件。为了简洁,上述语句也可以写成:

```
10 D$ =CHR$(4)
20 PRINT D$;"OPEN 学生"
```

或者写成:

```
20 PRINT CHR$(4);"OPEN 学生"
```

其中:CHR$(4)可以看成是嵌入 DOS 命令的标志,凡是涉及到文件,都必须有 CHR$(4)。OPEN 命令一定要用引号括起来并放在 PRINT 之后。

执行 OPEN 命令时需要完成的工作有:

①判断文件名是否存在。如果文件名已经存在则打开文件；如果文件名不存在则创建一个新文件，将文件名列入磁盘目录中。

②分配输入输出缓冲区。

执行 OPEN 命令后，DOS（磁盘操作系统）给文件分配 595 个字节的文件缓冲区，供内外存作为交换信息之用。

③确定打开文件的最大个数是否超界。

程序中一次能够打开文件的最大个数由 DOS 的 MAXFILE 命令来确定，至多不得超过 16 个。如果系统没有明确指定打开文件的最大个数，那么默认个数是 3 个。注意：

①在创立或检索文件之前，必须首先打开文件。

②OPEN 必须嵌套在 PRINT 之中。

2. 写文件的命令

格式：WRITE 〈文件名〉[,Bb]

作用：将内存的内容写入指定文件中。

其中：Bb—— 表示从顺序文件的第 b 个字节开始写数据。如果省略 Bb 待选项，那么就从顺序文件的第 0 个字节开始写入数据。

执行 WRITE 命令后，所有 PRINT 语句中输出的信息都输出到文件中而不是显示在屏幕上。因此，要写文件的话，WRITE 命令和 PRINT 语句一定要联用。

例 9.2　对学生文件输入数据。

```
10 D$ =CHR$(4)
20 PRINT D$;"OPEN 学生"
30 PRINT D$;"WRITE，学生"
40 PRINT "张 山 631 男 15 95"
…
```

20 句——创建学生文件；

30 句——写记录(或字段)到学生文件中去。

40 句——将“张 山 631 男 15 95”送到 学生文件中作为学生文件的第 0 个字段(本例将整个记录输入)。注意：

①WRITE 命令必须嵌套在 PRINT 之后且用引号括起来。

②WRITE 必须同 PRINT 联用。也就是说，WRITE 命令写入文件的数据是由 PRINT 传送的。没有 PRINT，数据就无法送到文件中去。

3. 关闭文件命令

格式：CLOSE [〈文件名〉]

作用：关闭指定的文件。如果没有指定文件名，就关闭当前打开的所有文件。

一个文件操作完毕后必须及时关闭。关闭命令至少有两个作用：一是关闭文件后使操作系统收回分配的缓冲区，用于其它急待打开的文件；二是文件关闭后能够防止可能的误操作影响到文件的数据。

下面我们用具体的例子来说明如何建立一个顺序文件。

例 9.3　用表 9.3 中的数据建立一个顺序文件。假设顺序文件的名字就用“学生”两个汉字。

程序如下：

```
10 D$ =  CHR$ (4)
20  PRINT D$;"OPEN 学生"
30  PRINT D$;"WRITE 学生"
40  FOR I = 1 TO 3
50  READ A$
60  PRINT A$
```

```
70  NEXT I
80  PRINT D$;"CLOSE 学生"
90  DATA  "张  山  631  男  15  95"
92  DATA  "李丽丽  632  女  13  88.5"
94  DATA  "王大力  634  男  14  74.7"
100  END
```

表 9.3　例 9.3 表格

姓名	学号	性别	年龄	成绩
张　山	631	男	15	95
李丽丽	632	女	13	88.5
王大力	634	男	14	74.7
…				

这就把表格中的数据送到“学生”文件中保存起来了。请注意观察，当运行程序时磁盘驱动器的指示灯在不断的闪亮，这就表明内存的数据在不断地送到磁盘上。如果你希望目睹程序的运行和数据的传送情况，那么在敲入 RUN 命令之前，请先敲入 MON C,I,O 命令。这就表示在执行程序的同时又在屏幕上显示出程序的执行过程。请自己试试。(程序最好保存在软盘上)。

注意:顺序文件只是保存了学生数据，并没有把例 9.3 的程序保存在其中，而程序文件才能用于保存程序，千万不能混为一谈。

9.3.2　顺序文件的检索

建立文件的目的主要是为了保存信息，便于检索。

1. 读入顺序文件

格式:READ 〈文件名〉 [,Bb]

作用:对打开的文件进行读入操作。在执行该命令时,所有INPUT 或 GET 语句都表示从顺序文件中接收数据而不是从键盘上接收数据。

其中:Bb——表示从顺序文件的第 b 个字节开始读入数据,如果省略了 Bb 待选项,则表示从顺序文件的第 0 个字节开始读入数据。注意:

①在用 INPUT 接收数据时,一次只能接收读入的一个字段值。如果字段值中有逗号,则依次将各数据送到 INPUT 后的各个变量中。

②如果 INPUT 中各变量的数据类型和顺序文件中接收的数据类型不相符合,那么就给出"? Reenter"(重新输入)信息,并继续接收下一个数据。

③如果顺序文件的字段中的数据多于 INPUT 语句中变量的个数,则给出信息:? EXTRAIGNORED(超过部分忽略不计),去掉多余部分。

④如果顺序文件字段中数据少于 INPUT 中变量的个数,屏幕显示??,系统自动到下一个字段中去读取数据。这种情况可能造成数据错位。

⑤为了防止数据错位,检索数据的格式和创建文件时的格式应该完全一致。

例 9.4　从顺序文件中读入数据。

我们以先前建立的学生文件为例,程序如下:

```
10 D$ =  CHR$ (4)
20  PRINT D$;"OPEN 学生"
30  PRINT D$;"READ 学生"
40  FOR I = 1 TO 3
50  INPUT A$
```

```
60  PRINT A$
70  NEXT I
80  PRINT D$;"CLOSE 学生"
90  END

]RUN
张  山   631  男  15  95
李丽丽   632  女  13  88.5
王大力   634  男  14  74.7
```

2. 添加数据

格式:APPEND 〈文件名〉[,Ss] [,Dd] [,Vv]

作用:打开顺序文件并允许在文件末尾添加新的数据。

其中:Ss,Dd,Vv 和前面含义相同。

注意:在 APPEND 之后必须用 WRITE 命令。

例 9.5　在学生文件末尾添加两个新记录数据。

```
10 D$ =  CHR$ (4)
20  PRINT D$;"APPEND 学生"
30  PRINT D$;"WRITE 学生"
40  PRINT "李  花  635  女  14  85.5"
50  PRINT "王  武  636  男  13  90.1"
60  PRINT D$;"CLOSE 学生"
70  END
```

通过运行以上程序,就把:

李　花　635　女　14　85.5

王　武　636　男　13　90.1

这两个新记录添加到学生文件的末尾了。学生文件原来只有三个记录,现在就有五个记录了。

3. 定位记录指针

格式：POSITION 〈文件名〉[,Rr]

作用：将文件的记录指针定位到指定的字段位置。

其中：Rr—表示把记录指针指向当前位置的第 r 个(包括 r)字段。如果省去了 Rr 待选项，就表示不移动指针。

注意：

①该命令必须作用于一个已打开的顺序文件。

②该命令必须用在 WRITE 和 READ 命令之前。

例 9.6　在学生文件中已经保存了五个记录数据(0～4 号)。现在希望从学生文件中读取第三、四、五个记录。可用下列程序实现(记住：记录号从 0 开始)：

```
10 D$ =  CHR$ (4)
20  PRINT D$;"OPEN 学生"
25  PRINT D$;"POSITION 学生,R2"
30  PRINT D$;"READ 学生"
40  INPUT A$,B$,C$
50  PRINT D$;"CLOSE 学生"
60  PRINT A$: PRINT B$: PRINT C$
70  END

]RUN
王大力  634  男  14  74.7
李  花  635  女  14  85.5
王  武  636  男  13  90.1
```

第 25 句将指针定位在第 2 号记录位置上(即 R_2 的作用)。

第 30,40 两句依次将第 2,第 3 和第 4 号记录送到字符串变量 A＄、B＄和 C＄中。

9.3.3 顺序文件的修改

严格说来，顺序文件是不能修改的。但我们可以利用 WRITE 命令来重写记录。

例 9.7 用“张 玲 637 女 14 93”去替换学生文件中的第一个记录。

```
10 D$ =  CHR$ (4)
20  PRINT D$;"OPEN 学生"
30  PRINT D$;"WRITE 学生"
40 A$ = "张  玲  637  女  14  93"
50  PRINT A$
60  PRINT D$;"CLOSE 学生"
70  PRINT D$;"OPEN 学生"
80  PRINT D$;"READ 学生"
90  INPUT B$
100  PRINT B$
110  PRINT D$;"CLOSE 学生"
120  END

]RUN
张  玲  637  女  14  93
```

通过执行 10～60 句，就用“张 玲 637 女 14 93”覆盖原来的 0 号记录。80～100 句就将替换后的新记录显示出来了。

对顺序文件，归纳起来必须注意以下几点：

①不论是建立、检索或修改文件都必须先打开(OPEN)文件(APPEND 命令除外)后使用。

②建立和修改文件时,OPEN、WRITE、PRINT 三者必须联用。(内存送外存)

③检索、打印文件记录时,OPEN、READ、INPUT 三者必须联用。(外存送内存)

④对于定位命令,如果定位是为了检索、打印记录,那么OPEN、POSITION、READ、INPUT 四者必须联用;如果定位是为了修改记录,那么 OPEN、POSITION、WRITE、PRINT 四者必须联用。

⑤文件使用以后必须关闭。

⑥一个新建的文件必须先关闭再重新打开后才能马上进行检索。不能够一边建立文件一边检索文件。

⑦顺序文件只能按顺序建立和顺序存取。

9.4 随机文件的建立和检索

9.4.1 随机文件的建立方式

1. 随机文件的建立命令

格式:OPEN 〈文件名〉Lj [,Ss] [,Dd] [,Vv]

作用:建立或打开一个随机文件。如果文件名不存在,就新建立一个文件并把文件名列入磁盘目录中;如果文件名已经存在,就打开该文件。

其中:

Lj——表示记录的长度。j 可以是 1～32767 之间的一个无符号整数。

Ss,Dd,Vv 含义同前述。

在用 OPEN 命令来建立随机文件时,DOS 并不把记录的长

度 Lj 存入磁盘中。如果打开文件时指定的记录长度与建立文件时的长度不一致，那么 DOS 始终按现在给出的记录来计算文件中各个记录的位置。如果存在前后记录长度不一的现象，读取记录时可能造成错位。为避免出错，创建文件时的记录格式和检索记录时的格式应当完全一样。

例 9.8　创立一个随机文件。

我们仍然以学生情况统计表格为例来说明如何建立随机文件。

```
10 D$ =CHR$(4)
20 PRINT D$;"OPEN 学生 L25,L25"
```

为了使随机文件名不同于顺序文件名，这里把随机文件的名字取为“学生 L25”，表示学生文件的记录长度是 25 个字符。L25 就表示了记录长度。

2. 写随机文件

格式：WRITE 〈文件名〉[,Rr][,Bb]

作用：将记录写到指定的随机文件中去。

其中：

Rr——表示指定文件的第 r 个记录，r 的值可以是 0~32767 之间的一个整数，这个数就表示把记录写到随机文件的第几号记录位置上。当 r=0 或省略不写时，表示从随机文件的第 0 号记录开始写入记录内容。从这里可以看到，随机文件并不象顺序文件那样必须顺序写入，而是可以按任何次序(由用户确定)来建立文件。当然，不按顺序来建立就必须指出记录号码，这就是参数 Rr 的作用。

Bb——表示从该记录的第 b 个字节开始执行写操作(即写入的内容从第 b 个字节开始存放)。如果 b 值太大以致超过了记录划定的长度，那么可能影响到其它记录的存放位置，从而导

致数据和记录位置错位。

在使用 WRITE 来写随机文件时，需要注意：

①使用 WRITE 之后，在没有关闭文件之前，不能用 DOS 的其它命令或 INPUT、GET 语句，否则，将会停止 WRITE 命令的执行。

②尽管随机文件记录的写入可以不按顺序，但我们最好还是依次顺序写入，以免弄错或弄掉一些记录。

例 9.9　随机文件的写入命令用法示例。

```
10 D$=CHR$(4)
20 PRINT D$;"OPEN 学生 L25,L25"
30 PRINT D$;"WRITE 学生 L25"
```

第 20 句创立一个随机文件"学生 L25"

第 30 句表示从随机文件的第 0 个记录位置开始写入记录。

在建立随机文件时，除了在 PRINT 中嵌套 OPEN 和 WRITE 命令以外，还需要有 PRINT 语句的配合，才能完成文件的建立工作。下面我们仍然使用学生情况统计表中的数据来建立一个完整的随机文件。

例 9.10　建立随机文件的完整示例。

```
10 D$ =  CHR$ (4)
20  PRINT D$;"OPEN 学生L25,L25"
30  FOR I = 1 TO 3
40  PRINT D$;"WRITE 学生L25,R";I
50  READ A$,B,C$,D,E
60  PRINT A$,B,C$,D,E
70  NEXT I
```

```
80  PRINT D$;"CLOSE 学生L25"
90  DATA  张  山,631,男,15,95
92  DATA  李丽丽,632,女,13,88.5
94  DATA  王大力,634,男,14,74.4
100  END

]MON C,I,O

]RUN
OPEN 学生L25,L25
WRITE 学生L25,R1
张  山          631              男              15
      95
WRITE 学生L25,R2
李丽丽          632              女              13
      88.5
WRITE 学生L25,R3
王大力          634              男              14
      74.4
CLOSE 学生L25
```

MON C,I,O 是监控命令,表示显示执行过程。清除监控用 NOMON C,I,O。

从以上显示的运行情况看出,我们需要的数据确实写入了随机文件"学生 L25"中。需要说明的是:

第 20 句表示打开随机文件,文件名字是"学生 L25",L25 表示随机文件的记录长度。打开文件必须按 20 句给出的形式书写。

第 40 句表示把数据写入文件。其中引号中的"R"表示记录

的意思，分号后的"I"用以标明是哪个记录（即记录号）。写文件的形式也必须如此。

第 50 句是将数据区的数据传送给变量 A $、B $、C $、D、E。

第 60 句才真正将数据传送到软盘上的文件中去。

第 90～94 句用于提供数据。从例中看到，字符型数据也可以不带引号。

要完成对随机文件的写入操作，也必须用 WRITE 配合 PRINT 语句。关闭文件的命令 CLOSE 总是需要的，格式同前述一样。

9.4.2 随机文件的检索

检索任何文件，均需要将其读入到内存，随机文件也不例外。现在看看随机文件的读入命令：

格式：READ 〈文件名〉[,Rr] [,Bb]

作用：将指定文件的指定记录读入内存。

其中：

Rr—— 表示对随机文件中的第 r 个记录执行读入操作。r 的值一定是随机文件的记录号。

Bb——表示从记录的第 b 个字节开始读入。

我们先来看下面的例子：

例 9.11　把学生 L25 随机文件的第 3 个记录读入到内存，并在屏幕上显示出来。

```
5  INPUT "I=";I
10 D$ =  CHR$ (4)
20  PRINT D$;"OPEN 学生L25,L25"
```

```
30  PRINT D$;"READ 学生L25,R";I
40  INPUT A$
50  PRINT D$;"CLOSE 学生L25"
60  PRINT A$
70  END

]RUN
I=3
OPEN 学生L25,L25
READ 学生L25,R3
?王大力634男1474.4
CLOSE 学生L25
 王大力634男1474.4
```

给出不同的I值就可以检索到文件中不同的记录。

9.4.3 随机文件的添加和修改

随机文件 的添加和修改都比较方便,我们要添加和修改哪个记录,只要给出记录号重新写入就行了。我们用例子来说明。

例 9.12 在学生文件中添加1个新记录。

4号 李花 635 女 14 85.5

程序如下:

```
10  INPUT "I=";I
20  INPUT A$,B,C$,D,E
30 D$ =  CHR$ (4)
40  PRINT D$;"OPEN 学生L25,L25"
```

```
50  PRINT D$;"WRITE 学生L25,R";I
60  PRINT A$,B,C$,D,E
70  PRINT D$;"CLOSE 学生L25"
80  END

]MON C,I,O

]RUN
I=4
?李  花,635,女,14,85.5
OPEN 学生L25,L25
WRITE 学生L25,R4
 李 花            635                女                      14
       85.5
 CLOSE 学生L25
```

例 9.13 将1号记录换成:“张 玲 637 女 14 93”

```
10  INPUT "I=";I
20  INPUT "输入修改数据:";A$,B,C$,D,E
30 D$ =  CHR$ (4)
40  PRINT D$;"OPEN 学生L25,L25"
50  PRINT D$;"WRITE 学生 L25,R";I
60  PRINT A$,B,C$,D,E
70  PRINT D$;"CLOSE 学生L25"
80  END

]RUN
I=1
 输入修改数据:张  玲,637,女,14,93
OPEN 学生 L25,L25
```

```
WRITE 学生L25,R1
 张 玲            637                 女                    14
        93
CLOSE 学生L25
```

我们要检查文件中的修改是否属实,可以编写一个检索程序(请读者自己完成),其运行结果如下:

```
RUN↙
张 玲     637   女   14   93
李丽丽    632   女   13   88.5
王大力    634   男   14   74.4
李 花     635   女   14   85.5
王 武     636   男   13   90.1
```

对于随机文件来说,我们要注意以下几点:

①建立文件、修改和添加记录时都是 OPEN、WRITE、PRINT 三者联用。

②检索文件时,OPEN、READ、INPUT 三者联用。

③无论是写(内存送外存)还是读(外存送内存)操作,均要提供记录号。机器就是根据记录号来进行直接存取的。当然,记录号并不要求一定依次连续,但不能超界。记录号的范围是 0~32767。在建立文件时,如果乐意的话,完全可以先写入 5 号记录,再写入 3 号记录。但这样交错进行的缺点是有时候自己都不清楚写入了哪些记录。因此,建议最好用顺序方式来建立文件,用随机方式来检索和修改文件。

④随机文件中不能用 POSITION 和 APPEND 命令。

⑤建立和检索文件之前用 OPEN 打开文件,然后用 CLOSE 关闭文件。

⑥所有命令都 必须嵌入 PRINT 之中才能生效。

问题与思考

1. 文件是若干数据的集合，数组也是若干数据的集合，两者的区别是什么？

2. 顺序文件和随机文件最根本的区别是什么？

3. 为什么随机文件不用 POSITION 和 APPEND 命令？

习题九

1. 分别用顺序文件和随机文件来建立如表 9.4 的人事文件：

表 9.4

代号	姓名	性别	年龄	职称	基本工资
84315	李顺	男	48	工程师	129.4
…	…	…	…	…	…

并对文件进行增加和修改，写出相应的程序。

2. 建立一个工资管理文件系统，该系统能够建立工资文件，增添新职工工资信息，修改职工工资，统计所有职工的总工资。写出进行这些操作的程序。

第二部分 LOGO 语言程序设计

本部分详细介绍了 LOGO 语言程序设计的方法和技巧。包括：LOGO 的基本概念，海龟直接作图，程序作图，数、字和表处理以及函数图形的编程方法，并给出了大量有趣图形的程序实例。

第十章　LOGO 语言的基本概念

10.1　什么是 LOGO 语言

LOGO 语言是美国麻省理工学院人工智能实验室于一九六八年开发的适合于青少年和初学者学习与使用的一种电脑语言。LOGO 语言有很强的作图功能，能够进行比较复杂的字和表的处理，同时也能够进行数值的计算和分析。

“LOGO”这个词，原意是“符号”或“思考”的意思。设计者用 LOGO 来作为一种电脑语言的名称，可谓用心良苦，含义深刻。LOGO 语言的主要发明者之一的佩帕特(Seymour papert)教授认为：刻板的计算机辅助教学，实际上是用程序设计去束缚青少年的想象力和创造力。六十年代对青少年进行计算机辅助教学的实践，也使人们认识到：辅助教学的目的，不应该是仅仅让计算机来教育青少年，而更重要的是要让计算机帮助青少年学会思考，提高分析和解决问题的能力。总之，应该使计算机成为青少年的朋友和助手，而不应该是他们的主人。

LOGO 语言采用青少年喜闻乐见的用积木拼图的方式，通过所谓的海龟作图来学习编制程序，训练他们的逻辑思维和创造能力，为进一步学习其它电脑语言打下基础。

佩帕特教授认为：计算机将会同电视一样普及，因而他们研制的 LOGO 语言不仅适用于青少年，也适宜于非计算机专业的人员学习和使用。

近几年来,LOGO 语言在国内已经有了较快的发展。国内使用的一般微型机,如 IBM PC,APPLE 及其兼容机等均配备有 LOGO 语言。目前流行的中华学习 CEC- I 上专门固化有 LOGO 语言的一个子集。

10.2　LOGO 语言的特点

1. 语句简单,易学、易用、易记

LOGO 的语句(又称命令)中,很多命令本身就是一个英文单词(部分命令还可以缩写),即使是不懂英语的人,记忆也不困难。目前还有拼音方式的 LOGO 语言问世,采用人们常用的汉语拼音书写的命令,更适合于青少年的学习使用。

2. 面向过程的语言

LOGO 的程序(又称过程)可以由一个或多个 LOGO 的基本命令组成,而这个过程一旦建立起来,又可以将它们作为用户自定义命令在程序中使用,因而 LOGO 有扩展性。

3. 有很强的会话能力

不论是 LOGO 的基本命令,还是由用户自定义的命令,都可以在键盘上直接输入,并被立即执行,执行的结果,也随即在屏幕上显示出来。当输入发生错误时,还会在屏幕上提示出错信息,使用户很方便的进行修改或重新输入。

4. 适宜模块化程序设计

一个大的程序可以由若干个小的程序组成,每一个小的程序又由 LOGO 的基本命令组成。这些程序可以独立使用,独立调试,独立修改,独立完成一定的功能,故称为模块。所有模块可以同时储存在计算机中,以备随时调用。

5. 允许过程嵌套和递归调用

当将某个过程模块作为自定义命令用于另外的过程之中时，称为“过程嵌套”，而将某过程模块作为自定义命令用于本身的过程中时，即自己调用自己，称为“递归”。过程的嵌套与递归功能，使编制高水平的结构化程序和绘制复杂图形变得更加容易。

6. 控制海龟移动实现作图

LOGO 可以直接控制海龟移动，使其在屏幕上留下移动的痕迹，从而完成绘图。直接控制海龟移动的命令既少又简单，而且还可以直接看到绘图的经过，因而青少年和初学者很容易掌握。

7. 具有数、字和表处理的能力

LOGO 提供了数、函数及字和表的处理功能，因而可以进行一些数值计算和字符处理。

10.3 LOGO 语言的启动

中华学习机 CEC-Ⅰ上可以使用几种版本的 LOGO 语言。固化在 CEC-Ⅰ主机上的 LOGO 语言，称为固化 LOGO。另外还有通过软盘来启动的 LOGO，如：APPLE-LOGO 和复合 LOGO 等。固化 LOGO 和软盘 LOGO 的启动方式略有不同。

1. 固化 LOGO 的启动

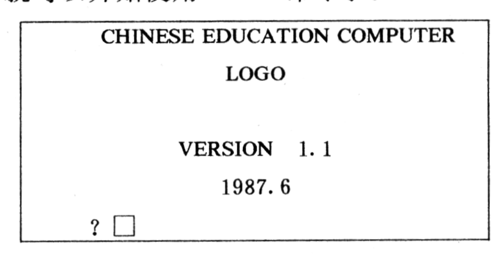
①按要求正确联接好硬件系统。

②打开显示器或电视机电源。

③稍候，再打开主机电源，系统自动进入西文 BASIC 状态，屏幕上出现提示符“]”和闪动的光标。

④键入 LG↙（用↙表示回车键，下同），屏幕显示如图 10.1。屏幕左下角出现 LOGO 提示符“?”和光标，至此，固化 LOGO

启动完毕，就可以开始使用 LOGO 命令了。


CHINESE EDUCATION COMPUTER

LOGO

VERSION 1.1

1987.6

? □

图 10.1 固化 LOGO 的启动

⑤若需进入彩色状态（外接必须是彩色显示器），可按下述步骤进行：当“?”提示符出现后，敲 **Ctrl+Reset** 键中断，进入监控，提示符变为“ * ”，然后再敲 **CTRL+Y** 键，则在屏幕左上角出现彩色提示符“?”，此时系统便进入彩色 **LOGO** 状态，这就可以使用彩色绘图命令了。还可以用命令进入彩色，这条命令是 .DEPOSIT 49328 16↙。

⑥若欲退出 LOGO，返回到 BASIC 状态，可使用 Ctrl+Reset 键中断，在屏幕上出现提示符“ * ”后，再使用 CTRL+B 键，然后回车，即可返回 BASIC，屏幕上出现“]”提示符。

另外，如果机器是用 DOS 来启动的，那么还可用下述命令来启动固化 LOGO：

]MAXFILE 1↙

]LG↙

这样，以后就可以对固化 LOGO 的过程和图形进行存盘和调用了。

2. APPEL-LOGO 软盘的启动

①主机启动前，将软盘按正确方向插入驱动器，关好驱动器的“门”。

②打开显示器电源。

③稍候，打开主机电源，驱动器即开始工作，屏幕显示如图10.2。

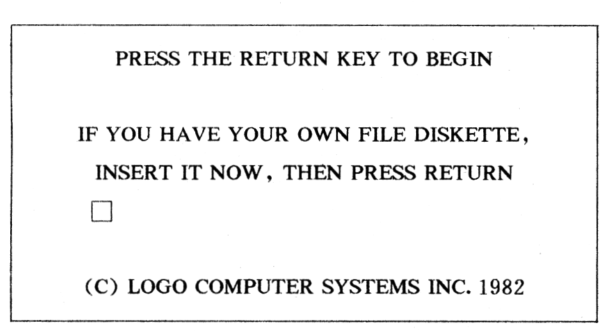


图 10.2　APPLE LOGO 的启动

屏幕提示的英文意思是："如果你有自己的 LOGO 文件盘，请把 LOGO 源盘取出，换上文件盘，然后敲回车键。若没有文件盘，敲回车键即可开始。"这是提醒用户：为 LOGO 源盘至少准备一个备份盘，以免当源盘遭受意外损坏后无 LOGO 系统盘可用。

④敲回车键后，稍候，屏幕显示如图 10.3。

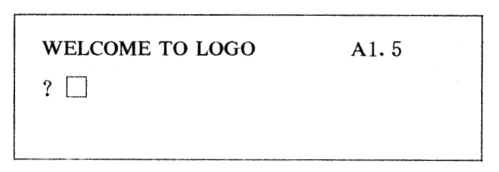


图 10.3　LOGO 启动后的屏幕提示

提示符"?"的出现，表明系统已进入 LOGO 状态(包括彩色状态)。

以上启动过程称为"冷启动"。

⑤若主机已经处于 BASIC 状态下，则在插入软盘后，在提示符"]"后面键入 PR # 6↙，则驱动器开始工作，同样可以进入 LOGO 状态。这称为"热启动"。

⑥若欲退出 LOGO，返回到 BASIC 状态，对于 APPLE-LO-

GO，一般用 CTRL＋RESET“中断”无效，可以用. BPT↙进入监控，再用 CTRL＋B 和回车键返回 BASIC 状态。

对于不同版本的 LOGO 软盘，启动过程和屏幕显示大同小异，可根据屏幕提示进行操作。

10.4 海龟的概念

1. 什么是海龟？

用 LOGO 语言来绘图，是通过控制所谓“海龟”的爬行来实现的。什么是海龟呢？

在“?”状态下，试键入 CS↙则清屏幕，并自动进入“图形/文本混合显示”状态，屏幕下方有四行文本区，供书写命令之用，而在屏幕中心位置出现一个白色的尖头向上的小三角形，这就是 LOGO 语言中的海龟，也有称其为“图龟”和“画龟”的。当海龟在有关命令驱使下移动时，其运动的轨迹即构成图形，这就是海龟作图。

海龟的形状，在固化 LOGO 状态下，可以用下述命令使之放大，放大后的形状如图 10.4。

? DRAW↙ (a)

?. DEPOSIT 230 2 DRAW↙ (b)

?. DEPOSIT 230 4 DRAW↙ (c)

?. DEPOSIT 230 6 DRAW↙ (d)

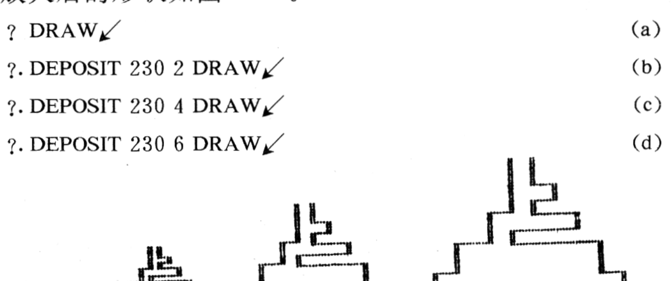


图 10.4 海龟的形状

正式使用海龟绘图时,若将海龟恢复原状,使用命令是
?. DEPOSIT 230 1 DRAW↙。

2. 海龟的移动

①前进:海龟按尖头方向爬行。

②后退:海龟按尖头反方向爬行。

③左转:海龟尖头按逆时针方向偏转。

④右转:海龟尖头按顺时针方向偏转。

在图 10.5 中,海龟最先处于原始位置(屏幕中心)。其中(a)图表示海龟前进了 50 步,(b)图表示海龟在(a)图位置上后退了 20 步,(c)图表示海龟在(b)图位置上左转 90 度,(d)图表示海龟在(c)图位置上又左转 120 度(也可以说是右转 240 度),(e)图表示海龟在(d)图位置上后退 20 步。

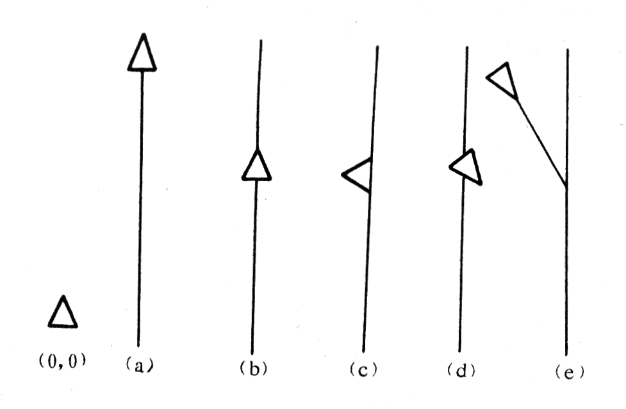


图 10.5 海龟的移动

3. 海龟的移动范围

屏幕取 X－Y 坐标如图 10.6,海龟的原始位置在屏幕中心,坐标为(0,0),纵横坐标的单位以海龟步计。

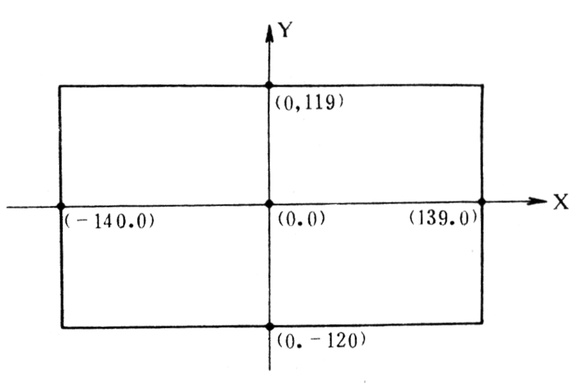


图 10.6　海龟的移动范围

横坐标 X 的范围是：$-140\leqslant X\leqslant 139$。

纵坐标 Y 的范围是：$-120\leqslant X\leqslant 119$。

为在屏幕上绘出正确的图形，必须使海龟在上述范围内移动。若海龟的活动范围超出上述界限，就会自动构成循环爬行，即海龟向上爬行超出 119 步(从原始位置算起)，它就会从对称的另一方向(屏幕下端)钻出来；而向左爬行超出 140 步，同样的也会从屏幕右端钻出来，反之亦然。这种循环爬行的结果，可能导致正确的图形被破坏，所以在使用绘图命令和设计程序时，应注意控制海龟在规定的范围内活动。

海龟的上述活动范围，是全图形显示时的情况。当屏幕处于混合显示方式时，系统将屏幕下方的四行留作文本区，所以海龟向下移动超过 80 步时，就看不见它的踪影了，但只要不超过 120 步，仍不会影响运行结果。

习题十

1. 按要求正确进行固化 LOGO 的启动，并使用 10.4 节所介

绍的命令，观察海龟的形状。(注意该命令中的空格，用空格键产生)。

2. LOGO 命令全部使用大写英文字母，请从键盘上按顺序键入 A、B、C、……Y、Z 26 个字符，及数字，注意观察字符 O、I 和数字 0、1 的差别。

3. 使用 APPLE-LOGO 软盘，练习“冷启动”，然后进入 BASIC 状态，再进行“热启动”。

第十一章　海龟直接绘图法

所谓直接绘图，就是使用基本绘图命令控制海龟的状态。每当敲入一条命令后，系统就立即执行此命令，海龟也就立刻移动或转动一次，从而在屏幕上完成绘图任务。即写则即绘，重绘必重写，命令不能保留。

当系统正确启动且屏幕上出现"?"提示符后，我们就可以用 LOGO 的基本绘图命令来直接作图了。

鉴于固化 LOGO 只能在中华学习机上使用，有一定的局限性。为此，本部分以 APPLE-LOGO 作为基础来介绍绘图、程序编制及计算等，其与固化 LOGO 的区别，将在相应的地方加以说明。

11.1　基本绘图命令

11.1.1　**移动和转向命令**

当我们在"?"后面键入 CS↙，屏幕进入图形/文本混合显示方式，屏幕中心出现海龟，我们就可以用海龟基本绘图命令来作图了。

1. 前进命令

格式：**FORWARD/FD**〈步数〉

作用：使海龟按指定的步数移动。当步数为正时，海龟沿尖头方向移动若干步，移动时，即留下轨迹；步数为负时，海龟沿尖

头反方向移动。注意：

①凡用“〈”和“〉”括起的内容(称参数)，使用时，只键入其中内容，而不键入“〈”和“〉”符号。

②命令与参数之间，至少用一个空格(敲空格键)隔开。

③FORWARD/FD 中，前者为全命令，后者为缩写形式，作用相同(以下类似)。

④海龟步数可正可负，也可以是小数，若使海龟按尖头方向前进时，步数必须为正。

2. 后退命令

格式：**BACK/BK** 〈步数〉

作用：当步数为正时，使海龟回退若干步，其作用刚好与 FD 命令相反。

3. 左转命令

格式：**LEFT/LT** 〈角度〉

作用：使海龟的尖头转动若干度，当角度为正时，海龟尖头按逆时针方向(左旋)转动，不移动；当角度为负时，海龟尖头则按顺时针方向(右旋)转动。注意：

①角度用“度”计，若为弧度值，需要换算。

②当角度值大于 360 度时，海龟转动的实际角度为减去 360 度的若干倍后剩余的度数。

4. 右转命令

格式：**RIGHT/RT** 〈角度〉

作用：与左转命令 LT 的作用相反。

例 1. 健入下面的命令，观察海龟的动作。

? CS↙　　(清屏、复位)

? FD 80↙　　(向上走 80 步)

? BK 50↙　　(后退 50 步)

? LT 90↙　　　　(原地左转 90°)

? RT 120↙　　　(原地右转 120°)

? FD 30↙　　　　(在上述位置,进 30 步)

以上命令的执行结果见图 11.1

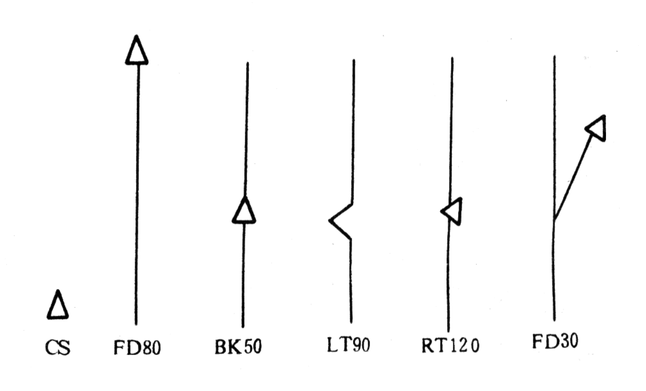


图 11.1　海龟的动作图示

例 2.　再来一次。

? CS↙

? FD 60↙

? RT 90↙

? FD 60↙

? RT 90↙

? FD 60 RT 90 ↙

(进 60,转 90°)

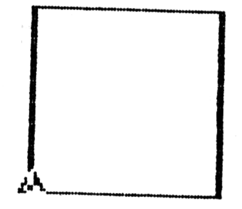


图 11.2　海龟画正方形

? FD 60 RT 90 ↙ (进 60,转 90°)

执行以上命令,在屏幕上得到一个边长为 60 的正方形,见图 11.2。

由以上结果，可以知道，通过使用移动和转向命令来控制海龟，只要改变命令所对应的参数值(步数或角度)，就可以获得一些简单的几何图形。当然也可以指使海龟任意移动和转向来获得随意图形。注意：

①命令前的“?”为提示符，不键入。

②“?”后可以输入多个命令，命令之间或命令与参数之间，必须用空格隔开。

③每行命令均以敲回车键结束，用“↙”表示。为简洁起见，以后的示例中一般不再给出，但仍然要记住敲回车键。

④使用 FD 或 BK 命令时，其步数应考虑屏幕范围，否则可能会造成图形错乱。

⑤当需要重新绘图时，最好将旧图清掉，可以使用 CS↙，CS为屏幕处理命令，见 11.3.1。

⑥要特别注意不要把数字“0”和字符“O”相混淆。

⑦若在输入命令或参数时，出现错误，屏幕将提示出错信息(见附录)，对于直接绘图，一般只需将正确的命令与参数按正确格式重新输入即可。

例 3.　利用上述绘图命令，绘一个边长为 60 的正三角形。

要想使图形顺利绘制成功，必须注意与其对应的几何关系，尤其是转动的方向和角度，必须搞清楚。为此目的，一般可以先在纸上画个草图，然后模仿海龟，由原始位置出发，沿着草图走一圈，记住每次走的步数和转过的角度，命令就可以写出来了。

草图如图 11.3。设原始位置定在三角形左下角处，且海龟尖头向上。

分析过程：

①先右转 30 度，

②前进 60 步，

③右转 120 度，

④前进 60 步，

⑤右转 120 度，

⑥前进 60 步，

⑦右转 90 度，回到原始位置。

按上述分析，敲入命令如下：



```
? CS
? RT 30
? FD 60 RT 120
? FD 60 RT 120
? FD 60 RT 90
```

敲入命令后，屏幕上即依次显示出一个正三角形，如图 11.4。

图 11.3　分析草图

读者不妨试用上述方法，绘出正五边形，正六边形以至更多边数的图形。

11.1.2　重复命令

从以上结果可以看出，有了移动和转向命令，一般图形都能够画出来了，但随着图形边数、边长的变化，及复杂程度的增加，单纯使用移动和转向命令，就显得不太方便了。我们分析一下前面绘制正方形的过程，不难看出，它实际上是由[**FD** 60 **RT** 90]这一组命令，重复四次而完成的，当我们使用下述重复命令时，将使绘图命令大大简化。

格式：**REPEAT** 〈次数〉[〈命令〉]

作用：将方括号中的命令重复执行指定的次数。

例 4. 对于前述绘制正方形的过程，用重复命令实现，则为：

```
? CS
```

? REPEAT 4[FD 60 RT 90]

计算机执行此命令,将使海龟在屏幕上连续绘出一个正方形。注意:

①重复次数必须是正整数。

②待重复执行的命令必须用方括号“[”和“]”括起来。(在 APPLE 机上,左方括号敲 SHIFT + N 键,右方括号敲 SHIFT+M 键)。

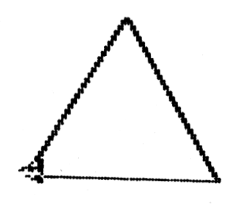


图 11.4 正三角形

读者不妨用重复命令,绘出正五边形、正六边形和多边形,也可以将命令序列中的 RT 换成 LT,再试试看,结果如何。

11.1.3 抬笔和落笔命令

细心的读者可能发现,以上绘制的图形都是从海龟的原始位置开始的,而且图形都是实线,那么若想在坐标范围内的任意位置开始绘图,或者准备绘制虚线组成的图形时,该怎么办呢?

若将海龟移动画线看成是一支画笔在绘图的话,那末画图时,画笔是落在纸上的;不需画图时,把笔抬起来;换个地方再画时,只要把笔再落下来,问题不就解决了吗?LOGO 中提供了抬笔和落笔命令。

1. 抬笔命令

格式:**PENUP/PU**

作用:键入此命令,海龟状态不变,但当用移动命令使海龟移动时,屏幕上不会显示轨迹,相当于把画笔抬起。

2. 落笔命令

格式:PENDOWN/PD

作用:画笔在 PU 命令下抬起后,若再键入此命令,则海龟再移动时,又会留下轨迹,相当于将抬起的画笔重新落下。注

意：

①PU、PD 命令后均无参数。

②PU 和 PD 命令必须成对使用。

例 5. 画出中心在(－50,50)处的边长为 80 的正方形。

```
? CS                           (清屏、复位)
? PU                           (抬笔)
? FD 10 LT 90 FD 10 RT 90      (调动海龟)
? PD                           (落笔)
? REPEAT 4[FD 80 LT 90]        (画正方形)
```

例 6. 画出中心在(50,50)处的半径为 40 的圆(以 36 边形代替)。

```
? CS
? PU
? FD 50 RT 90 FD 10 LT 90
? PD
? REPEAT 36[FD 40*3.14/18 RT 10]
```

以上二例结果，分别见图 11.5 和图 11.6。

表达式“40 * 3.14/18”是为了保证圆的半径为 40 而计算的步数值。

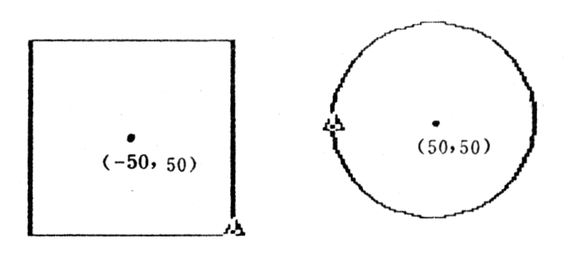


图 11.5 正方形　　图 11.6 圆

例 7.　画出中心在原点，半径为 80 的虚线圆，要求虚线间隔相等，共 36 条。

```
? CS
? PU FD 80 RT 90 PD (抬笔、调动、落笔)
? REPEAT 36[FD 80 * 3.14/36 RT 5 PU FD 80 * 3.14/36 RT 5 PD]
```

所得图形见图 11.7。

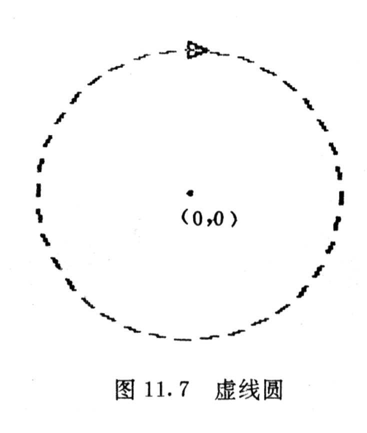


图 11.7　虚线圆

注意：当命令中的字符(包括空格)占满一行时，系统会自动换行，同时在原行末尾出现“！”提示，表示为系统自动换行。书写程序时，为与敲回车键而换行相区别，自动换行后的命令退后二个以上字符位置书写。当需要自动换行时，不能敲回车键，只要将字符依次继续输入即可。

11.1.4　坐标绘图命令

从上面的示例可以看出，使用前述命令来调动海龟(包括多次移动和转向)是比较麻烦的，因为在调动的过程中，需不断调整海龟的运动方向，尤其是若不知海电的当前方向，极易出错。为此下面介绍按坐标位置来移动海龟的方法。

1. 横坐标命令

格式：**SETX** 〈X 坐标〉

作用：将海龟按水平方向移到 X 坐标值的位置，Y 坐标位置维持不变。

2. 纵坐标命令

格式:**SETY** 〈Y 坐标〉

作用:将海龟按垂直方向移到 Y 坐标值的位置,X 坐标位置维持不变。

3. 纵横坐标命令

格式:**SETPOS** [〈X 坐标〉〈Y 坐标〉]

(固化 LOGO 命令为:SETXY 〈X〉〈Y〉)

作用:将海龟移到指定的坐标位置。

4. 纵横坐标打点命令

格式:**DOT** [〈X 坐标〉〈Y 坐标〉]

(固化 LOGO 无)

作用:在坐标(X,Y)处打一点,与海龟无关。

当执行坐标绘图命令时,海龟将由当前位置向坐标位置直接移动(与海龟方向无关),移动过程若在落笔命令之后,则将留下轨迹,从而完成绘图。注意:

①坐标值以海龟步计,可正可负,也可为小数,坐标范围见图 11.6。

②使用 SETPOS 命令时,X 和 Y 坐标值以空格隔开,外面必须用方括号"["和"]"括起来。(固化 LOGO 命令 SETXY 后不用方括号,但 X 和 Y 值中为负时,需用圆括号"("和")"括起来)。

例 8.　仍画出中心在(-50,50)处的正方形。

```
? CS
? PU SETPOS [-10 10]PD (抬笔、调动、落笔)
? REPEAT 4[FD 80 LT 90]
```

请将此例中的调动命令与例 5 相比较。

11.2 重复命令的嵌套使用

如果我们要画一个如图 11.8 所示的风车图形，该怎样考虑呢？从图上不难看出，它是由一个长方形每次旋转 90 度，经过四次画成的，显然这也是一个重复的过程。

例 9. 用嵌套的重复命令画一个风车图案。

```
? CS
? REPEAT 4[REPEAT 2[FD 60 RT 90
          FD 30 RT 90]RT 90]
```

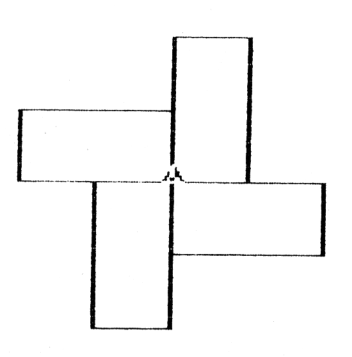


图 11.8 风车图

在上述命令行中，里面的 REPEAT 命令画出长方形，外面的 REPEAT 命令使之旋转四次，每次右转 90 度，即得图 11.8 风车图形。注意：

①方括号必须成对使用。

②命令嵌套可以有若干层，系统执行时，先内后外，逐级进行。

③欲使所绘图形封闭，海龟转动的角度总和应为 360 度的整倍数。

下面再给出几个使用嵌套命令绘图的示例 。

例 10. 旋转三角形。

```
? CS
? REPEAT 5 [REPEAT 3[FD 50 RT 120]RT 72]
```

结果如图 11.9。

例 11. 画一朵花。

```
? CS
? REPEAT 8[REPEAT 2[REPEAT 6[FD 12 RT 10]RT 120]RT 45]
```

结果如图 11.10。

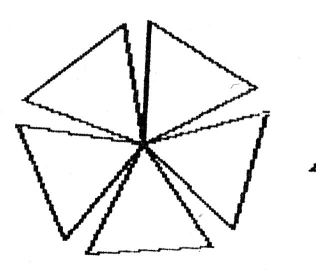
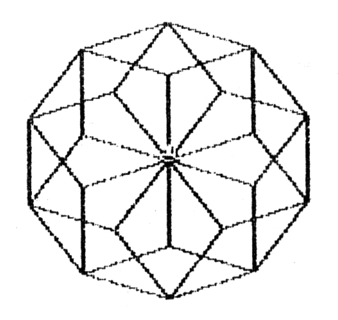


图 11.9　三角形　　　　图 11.10　小花

例 12.　旋转五边形。

```
? CS
? REPEAT 10[REPEAT 5[FD 50 RT 72]RT 36]
```

结果如图 11.11



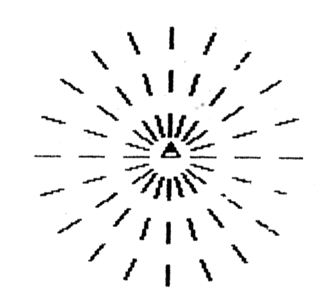
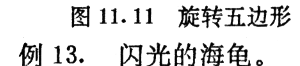


图 11.11　旋转五边形　　　　图 11.12　闪光的海龟



例 13.　闪光的海龟。

```
? CS
? REPEAT 20[REPEAT 3[PU FD 12 PD FD 12]
```

PU BK 72 PD RT 18]

结果如图 11.12。

11.3 其它命令

11.3.1 屏幕处理

1. 海龟消隐

格式:**HIDETURTLE/HT**

作用:HT 命令使屏幕上只显示海龟的移动轨迹,而不再显示出海龟,相当于将海龟隐藏起来。如果在绘制图形时不希望出现海龟,则用此命令。使用此命令,还可以提高绘图速度。

2. 海龟重显

格式:**SHOWTURTLE/ST**

作用:ST 命令可以使消隐的海龟重新显示在屏幕上,便于了解海龟的当前位置。

3. 清屏复位

格式:**CLEARSCREEN/CS** (固化 LOGO 命令为:DRAW)

作用:清除屏幕图形(不清内存),使屏幕自动处于图形/文本混合显示方式,海龟回到原始位置。此命令一般用在绘制新图之前对屏幕进行清屏处理。

4. 复位不清屏

格式:**HOME**

作用:保留屏幕图形,仅使海龟返回到原始位置(在 PD 命令后,返回时,要留下轨迹),多用于程序间的处理。

5. 清屏不复位

格式:**CLEAN** (固化 LOGO 命令为:CS)

作用:只清屏幕,海龟留在当前位置,状态不变。

11.3.2 测定海龟的当前状态

1. 测定角度

格式:**HEADING**

作用:测定海龟当前位置的方向与 Y 轴之间所形成的夹角的大小(均以顺时针偏离 Y 轴的角度计算)。

例 14. 测定偏移角度。

```
? CS
? REPEAT 30[FD 3 RT 2]        (绘图)
? PR HEADING                  (测定角度)
60                            (结果)
```

结果表明海龟当前方向沿顺时针偏离 Y 轴 60 度。其中"PR"为打印输出命令,见第十三章。

2. 改变海龟方向

格式:**SETHEADING/SETH** 〈角度〉

作用:将海龟的方向按顺时针偏离 Y 轴指定角度,与其原方向无关。

3. 坐标位置测定

格式:**XCOR/YCOR**

作用:测定海龟当前位置的横/纵坐标值。

例 15. 测定海龟的坐标位置。

保持例 14 的图不变(图 11.13)。再键入:

```
? PR XCOR         (测定横坐标)
41.6684           (结果)
? PR YCOR         (测定纵坐标)
75.1717           (结果)
```

结果表明海龟停留处的坐标为:41.6684,75.1717。

上述对应的图形、角度及坐标如图 11.13 所示。

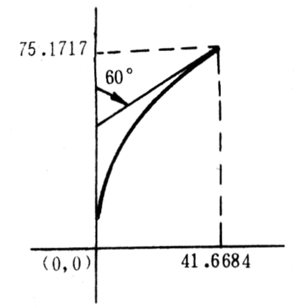


图 11.13 海龟的偏移角度

11.3.3 图形的纵横比例设置

通过实际上机操作，读者可能已经发现，屏幕上绘出的图形，出现正方形不方，圆形不圆的情况，这是什么原因呢？这是由于屏幕的水平方向和垂直方向距离长度的比例不等而造成的。即是说同样一步，水平方向移动比垂直方向移动的距离要长一些。为了减小由于比例不当引起的图案变形，LOGO 提供了图形纵横比例设置命令。

格式：**SETSCRUNCH 〈X〉**

（固化 LOGO 的命令为：. ASPECT 〈X〉）

作用：改变参数 X 的值，可以改变屏幕纵横比例的关系，X 值的设置范围为 0.6～1.4。

当 LOGO 正确启动后，图形比例参数自动设为 0.8，其值越小，图形越扁，本书中的部分插图和附图都是在 X 取值为 1.15 时打印下来的。当 X 的取值增大时，对应的 Y 坐标的允许范围将相应减少，作图时必须注意。一旦给定 X 值，关机前的绘图则均在此比例下进行，若欲恢复开机状态，再执行一次此命令，并使 X 取值为 0.8 即可。

读者可以用画圆来试验，当给出 X 的不同取值时，观察图形的差别。

以上介绍了 LOGO 的基本绘图命令。在使用时经常涉及到屏幕的显示方式，不同显示方式用以下控制键实现：

Ctrl＋L：　全图形显示，即显图隐字。

（固化 LOGO 控制键为：Ctrl＋F）

Ctrl＋T： 全文本显示，即显字隐图。

Ctrl＋S： 图形/文本混合显示，即图形文字双显。

习题十一

1. 用重复命令绘出五边形和六边形。

2. 用重复命令绘出五角星。

3. 通过原点，沿水平方向从－120 到 120，画一条点划线作为 X 轴：线段长为 26，间隙为 1，点长为 2。

4. 通过原点，沿竖直方向从－90 到 90，画一条虚线作为 Y 轴：线段长为 10，间隙为 10。

5. 画出图 1 所示的旋转正方形。

6. 画出图 2 所示的正方体（尺寸自定）。

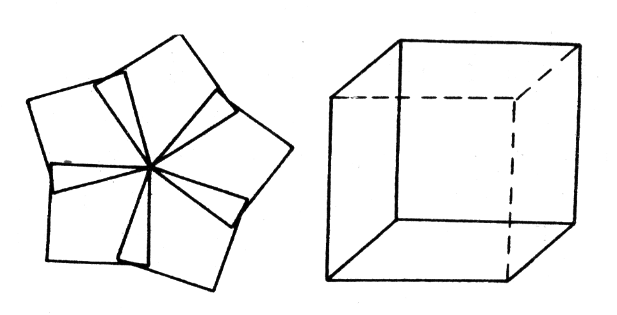


图 1 旋转正方形　　　　图 2 正方体

7. 画出图 3 所示的表格。

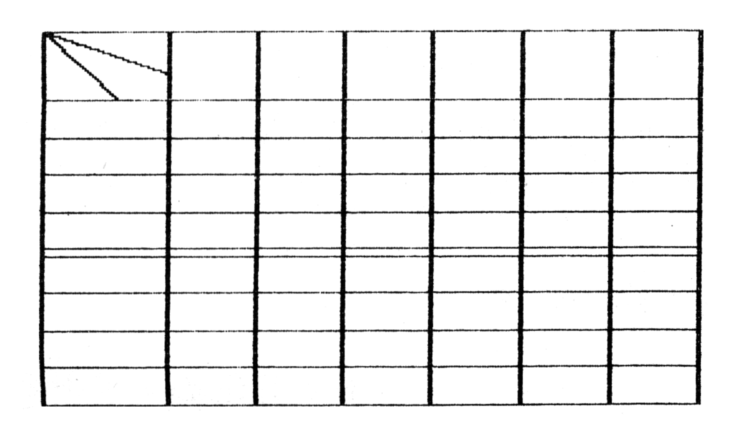


图 3　表格

第十二章　海龟程序绘图法

第十一章介绍的基本命令直接绘图的方法虽然简单，却存在以下缺点：一是敲入的命令只能使用一次，不能保存。二是命令涉及的参数都是固定的，不能随具体情况的不同而发生变化。为克服这些缺点，LOGO 又提供了过程绘图。本章将要介绍如何用过程来实现绘图。

12.1　过程及其建立

12.1.1　什么是过程

所谓过程，简单说来，就是由若干基本命令按一定要求汇集在一起且能够完成一定功能的一个程序段。程序段在建立时，键入的各条命令并不立即执行，只有等程序建立完毕，并被系统承认以后，才能用专门的标志来调用。程序内的命令才得以按先后次序逐条执行。程序段一旦建立，用户就可以象 LOGO 命令一样来使用这种自定义命令。这样的程序段，通常称之为“过程”。

建立过程时，必须向系统提供以下信息：

1. 定义过程的标志是什么

该标志应表明用户开始定义过程，系统对随后键入的各项命令不再立即执行。

2. 过程的名字是什么

过程的名字就是程序段的总代表。它可以由字母或数字组

成，但必须以字母开头。过程名中不能含有空格、括号或运算符。过程名也不能与 LOGO 命令相同，多个过程彼此不能同名。

3. 是否提供过程参数

当过程需要同外界交换信息时，就必须提供过程参数。参数可以是一个或多个，它的取名方法同过程名一样。不同名的过程中参数名可以相同，同一过程中各参数不能同名。建立过程时提供的参数称为形式参数，简称形参。调用过程时给出的参数称为实在参数，简称实参。

4. 过程体包含哪些命令

这是过程中最主要的部分，完成特定的操作功能就靠过程体。因此，要用到的全部命令都必须在过程体中给出。

5. 过程的结束标志

过程定义完毕，通常应给出结束标志，以示该过程到此为止。

12.1.2 过程的结构与建立方法

1. 过程的结构

格式：**TO** 〈过程名〉〈参数 1〉〈参数 2〉……〈参数 n〉

〈过程体〉

END

作用：允许建立一个过程。

TO：定义过程的标志。

过程名：用户为过程取的名字。注意取名要便于记忆，一般有两种方案：一是用该过程功能的英文单词或缩写来命名，二是用汉语拼音或拼音字头来表示，当然也可以任取一个名字。

参数：根据需要而定，可有可无，可多可少。没有参数的过程称为“无参过程”，无参过程只能完成单一的绘图、计算或分析工

作。带参数的过程称为“有参过程”，参数值的改变，有时会影响过程的功能。

过程体：通常是为完成某种操作的一组 LOGO 命令。当然可以包括定义了的其它过程（即过程嵌套），还可以包含过程本身（即过程递归）。

END：过程定义完毕的结束标志。

2. 过程的建立方法

在键入 TO〈过程名〉〈参数 1〉〈参数 2〉……后（若没有参数，只需键入过程名），敲回车键，则提示符由“?”换成“〉”，表示系统进入过程定义状态，这时就可以键入过程体中的各条命令了（此后，每行的提示符均为“〉”）。（固化 LOGO 过程建立与编辑相同，见 12.5.2）。

当命令输入完毕，换行后键入 END，再敲回车键，屏幕显示出“〈过程名〉 DEFINED”字样，表示过程已经定义完毕，随之提示符变为“?”，表示定义的过程已被系统承认。这样，此过程就可以象 LOGO 命令一样随时调用，相当于系统中增加了一条新的命令。

12.2 无参过程和有参过程

12.2.1 无参过程

仍以绘制正方形为例。

例 1. 用无参过程来编写绘制正方形图形的程序。这里只需要将前面的命令序列“戴帽穿靴”即可。

TO DB　　　　　　　　（“戴帽”）

〉CS

```
>PU SETPOS [-10 10]PD
>REPEAT 4[FD 80 LT 90]
>END                      (“穿靴”)
```

当键入 END↙后，屏幕显示：“DB DEFINED”字样，表示以“DB”命名的过程已定义，并被系统承认，然后出现提示符“?”。

在“?”后面键入 DB↙，系统将调用 DB 过程，并按该过程体中命令的顺序依次执行：清屏复位→海龟调动→画正方形→停止。从而在屏幕上连续地、自动地画出一个正方形。绘图完毕，提示符出现在文本第一行(混显方式)。

若再次键入 DB↙，则系统又执行一次上述过程，画出同一个正方形。显然，过程一经定义，根据需要可随时多次调用，这相当于在 LOGO 的命令中增加了一个在固定位置绘制固定长度的正方形图形的专用命令，这个命令的名字就叫“DB”。

例 2. 编写一个绘制五角星的过程。五角星的几何关系见图 12.1。

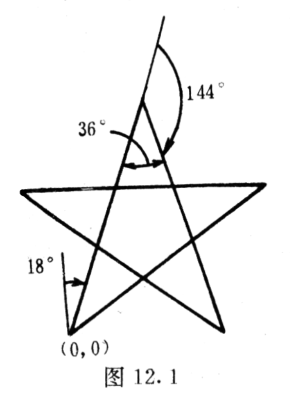


图 12.1

```
TO W
>CS RT 18
>REPEAT 5[FD 60 RT 144]
>END
```

定义完毕，键入 W↙，屏幕上就自动绘出一个五角星，如图 12.2。

从以上二例可以看出，用无参过程只能绘制固定边数、固定边长和固定位置的固定图形。若欲绘制能够变化长度、位置等的图形，势必要同时定义相当多的无参过程，这显然是很不方便的，甚至是不可能的。

12.2.2 有参过程

所谓有参过程，即是把过程中各项命令的某些参数用形式参数来表示，置于过程名之后。当执行过程时由用户临时输入该参数的值，这样当输入的值不同时，得到的结果就不一样，从而大大地扩充了过程的功能。

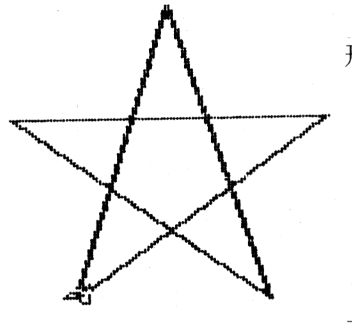


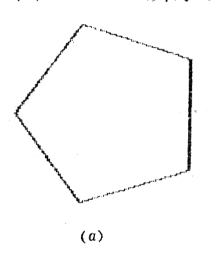
图 12.2

例 3. 编写能够绘制各种正多边形的有参过程。

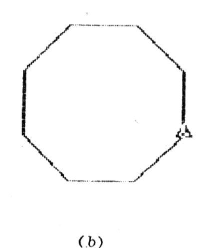
```
TO DB1 :A :B
〉CS
〉PU SETPOS [-10 10] PD
〉REPEAT :A[FD :B LT 360/:A]
〉END
```

过程"DB1"定义完毕，其中 :A 表示正多边形的边数，:B 表示边长。在调用这个过程时，必须以实在参数替换过程中的形式参数。如：

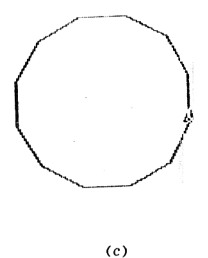
? DB1 5 60↙ 屏幕上显示出一个边长为 60 的正五边形，如图 12.3(a)所示。



(a)



(b)



(c)

图 12.3 绘制各种正多边形

? DB1 8 35↙ 屏幕上显示出一个边长为 35 的正八边形，

如图 12.3(b)所示。

? DB1 12 25↙ 屏幕上显示出一个边长为 25 的正十二边形,如图 12.3(c)所示。

由此可见,给出不同的参数值,就能得到不同的正多边形。读者不妨自己试一试。

LOGO 的过程具有很大的魔力,只要稍稍改变一下参数的实际值,就可能获得意想不到的效果。但在使用有参过程时,要注意:

①形式参数之前一定要加冒号":",而且冒号与参数之间不能留有空格。

②过程名与形参,形参与形参之间都必须用空格隔开。

③当执行有参过程时,实参与形参在个数、位置和顺序上必须一一对应。替换时,实参前不用冒号":"。

上面的有参过程是在指定位置上开始绘图的,若欲在任意位置开始绘图,可增加变动位置的参数。

例 4. 在任意位置开始绘制正多边形。

```
TO DB2 :X :Y :A :B
>CS
>PU SETPOS SE :X :Y PD
>REPEAT :A[FD :B LT 360/:A]
>END
```

此过程与过程 DB1 的区别在于运行时必须用四个实参来代替对应的形参,其中:X、:Y 为起始点的位置坐标。如:

? DB2 −50 0 8 35↙

注意观察绘图过程及其与 DB1 绘图的区别。

"DB2"过程体中出现的命令"SETPOS SE"。可当作另一条纵横坐标绘图命令使用。此命令后面的二个参数至少有一个是

形参，而且二个参数外面不再用方括号，这是由 APPLE-LOGO 本身的特点决定的，固化 LOGO 无此规定。使用时注意 SETPOS 与 SE 之间必须用空格隔开。同样，对于纵横坐标打点命令，也如此处理。

12.3 过程的嵌套

1. 无参过程的嵌套

第十一章中用到了重复命令，如果把每一个重复命令中的重复部分写成一个过程，那末重复命令的嵌套，即变成了过程的嵌套。

例 5. 编写绘制第十一章介绍过的长方形和风车图形的过程（为简洁起见，过程定义提示符“〉”略去）。

建立绘制长方形的过程：

```
TO CF1
REPEAT 2[FD 80 RT 90 FD 40 RT 90]
END
```

建立绘制风车的过程：

```
TO FC1
REPEAT 4[CF1 RT 90]
END
```

定义完毕，运行过程“CF1”，可得一长方形，运行过程“FC1”可得风车图形。

在过程“FC1”中，用到一个新命令“CF1”，它实际上是一个画长方形的过程名。

过程一旦定义，就可以当作一条命令使用，这条命令具有它所定义的过程体中所包含的全部基本命令功能，既可以独立使

用，也可以被调用。

例 6. 绘一个正在“转动”的风车。

```
TO ZD
FC1
PU FD 95 PD RT 90 BK 6
REPEAT 4[H PU H PD]
END
```

此过程中尚嵌有一个新过程“H”。

```
TO H
REPEAT 5[FD 95*3.14/20 RT 9]
END
```

我们又建立了过程“H”和过程“ZD”。在过程“ZD”中嵌有“FC1”和“H”二个过程，而过程“FC1”中还嵌有过程“CF1”。请运行：

? FC1↙　　　　运行结果如图 12.4

? CS ZD↙　　　运行结果如图 12.5

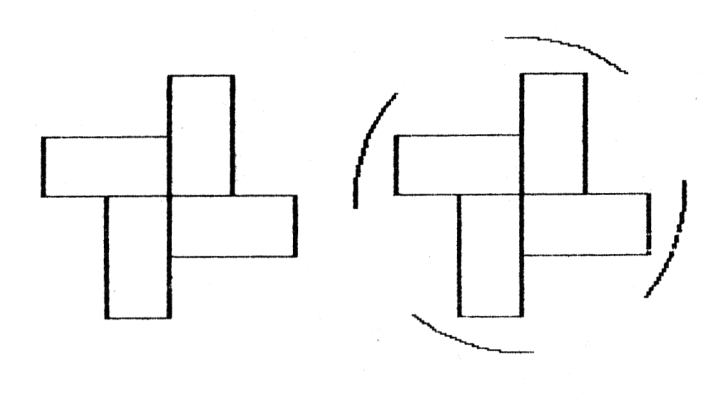


图 12.4 风车图　　　　图 12.5 旋转的风车

注意：

①过程嵌套可以多层，系统在执行过程时，依次按顺序运行。当遇到嵌套的过程时，就去执行嵌套的过程，执行完毕再返回继续依次向下执行。

②嵌套的过程，必须先定义，后使用。

无参过程的嵌套，虽然使绘图功能大大提高了一步，但仍然只能绘制固定的图形，为此，下面再介绍有参过程的嵌套。

2. 有参过程的嵌套

有参过程的嵌套与无参过程的嵌套相似。

例 7. 编写绘制长方形和风车图形的有参过程。

```
TO CF2 :M
REPEAT 2[FD  :M RT 90 FD  :M/2 RT 90]
END

TO FC2  :M  :N
REPEAT  :N[CF2  :M RT 360/ :N]
END
```

上述有参过程中的形式参数：M 表示长方形的长，：N 表示长方形的旋转的次数(即风车叶片数)。请运行：

? FC2 60 3↙　　　运行结果见图 12.6(a)。

? CS FC2 60 5↙　　运行结果见图 12.6(b)。

? CS FC2 60 8↙　　运行结果见图 12.6(c)。

? CS FC2 60 12↙　 运行结果见图 12.6(d)。

注意：运行时，实在参数的个数和位置必须和过程的形式参数一一对应。

例 8. 编写绘制带圆弧的图形。

```
TO YH  :A  :B
REPEAT  :A[FD  :B RT 10]
END
```

```
TO HY1 :A :B :C :D
REPEAD :C[YH :A :B LT :D]
END

TO HY2 :A :B :C :D
REPEAT :C[YH :A :B REPEAT :A[LT 10 BK :B]LT :D]
END
```

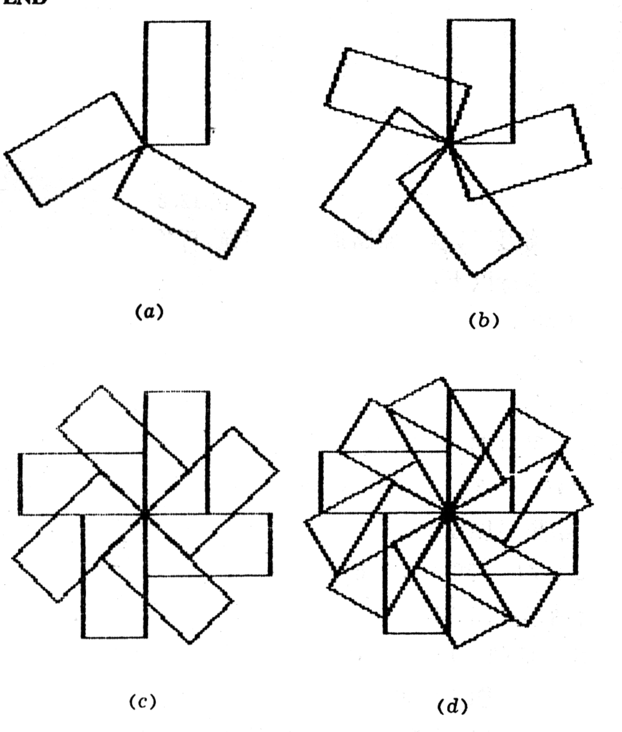


图 12.6 长方形和风车的旋转

过程“YH”用来画圆弧，当：A＝36 时，为圆形。过程“HY1”用来画旋转圆形，：C 为旋转圆（圆弧）的个数，：D 为每次旋转的度数。过程“HY2”用来画旋转圆弧，与过程“HT1”的区别是，

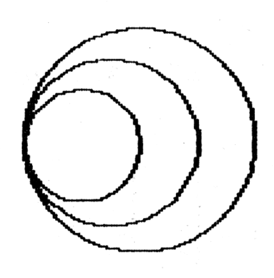


图 12.7　　　　图 12.8

每画完一段圆弧后，要沿原路退回到原地，再旋转。

试运行以上过程：

? YH 36 10↙

? YH 36 8↙

? YH 36 6↙

运行结果见图 12.7

? CS HY1　30　6　3　60↙

运行结果见图 12.8。

? CS HY1　36　5　6　60↙

运行结果见图 12.9。

? CS　HY2　14　5　12　30↙

运行结果见图 12.10。

以上过程，仅仅改变了某些参数的值，就使得绘出的图形变得多姿多态，已足见 LOGO 的绘图能力了。如果再用上 LOGO

过程的递归功能，才会真正显示出它的神奇魔力呢！

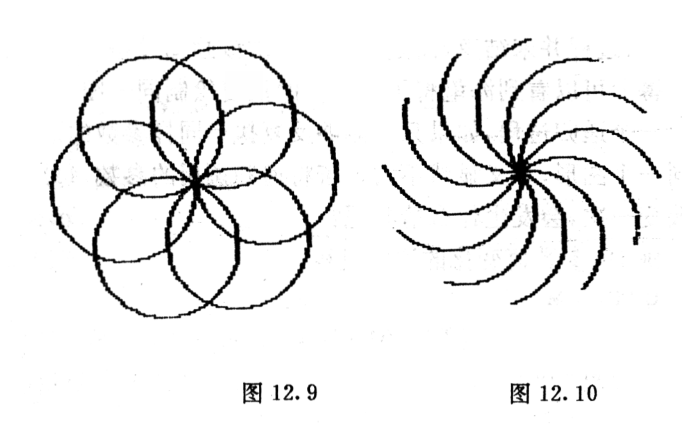


图 12.9　　　　图 12.10

12.4　递归过程

12.4.1　什么叫递归

我们已经知道，一个过程可以调用另一个过程，那末过程能否调用自身呢？当然可以。这种在过程体中又出现调用自身的情况，称为过程的递归。如果一个过程段中调用了自身，这称为过程的递归调用。允许递归调用的过程，称为递归过程。

递归过程的建立方式很简单，就是在有参过程结束前(即END之前)，将该过程名及所带的全部形式参数重写一次即可。

例 9. 画长方形的递归过程。

```
TO CF3 :M
REPEAT 2[FD :M RT 90 FD :M/2 RT 90]
CF3 :M                    (递归调用)
```

```
END
```

现在来看这个递归过程的执行情况：

运行开始，先执行过程体中的第一条命令—画出长方形，长方形画完后，过程并未结束，而是又执行一次该画图过程，如此反复。从屏幕上可以看到海龟将不知疲倦的反复绘制同一图形。

一个递归过程，如果只是能够反复执行同样参数的过程，画出同一个图形，意义显然不大。若将递归调用的参数，每次调用时改变一次，会发生什么情况呢？

例 10. 参数值变化的递归过程。

```
TO CF4 :M
REPEAT 2[FD :M RT 90 FD :M/2 RT 90]
CF4 :M+10
END
```

在此过程体中，与过程 CF3 不同的是其形式参数 :M 在递归调用时加上了一个增量。当过程第一次运行时，其对应的参数为 M(:M 的实在参数值)，而第二次运行时，对应的参数是 :M +10，即将原输入的实在参数值自动地加上增量，然后再重复执行此过程。键入：

? CF4　30↙

从运行可以看到，每运行一次，长方形的边长就长大一点(即增量 10)，结果如图 12.11。

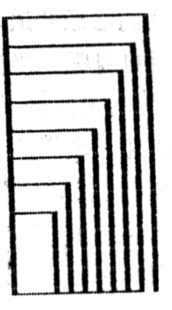


图 12.11

显然这种运行不会停止。欲让运行停止，可以敲 Ctrl＋G 键，强行使之停止，也可以在过程中使用条件语句(见 12.4.3)令其停止。

如果我们在上述过程中，再加入一

个转向命令，就会发现，我们所绘的长方形将会一边长大，一边旋转。

例 11. 绘制正在长大和旋转的长方形。

```
TO CF5 :M
REPEAT 2[FD :M RT 90 FD :M/2 RT 90]
RT 30
CF5 :M+10
END
```

试键入：

? CF5 20↙

运行结果见图 12.12。

由以上实例可知，实现过程的递归功能，就是在过程体结束前的适当位置，调用自身，并可改变某些形式参数的值。

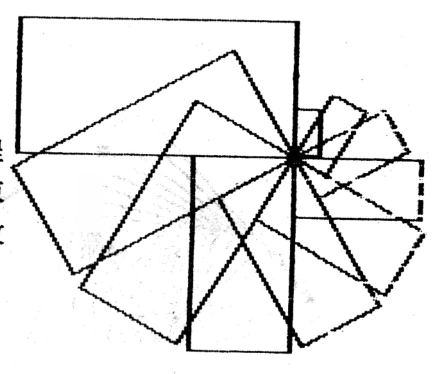


图 12.12 长方形的长大和旋转

12.4.2 递归过程实例

例 12. 旋涡线。

```
TO XW1 :A :B
FD :A RT :B
XW1 :A+3 :B
END
```

运行此过程：

? CS XW1 0 90↙，　　见图 12.13(a)。

? CS XW1 0 72↙，　　见图 12.13(b)。

? CS XW1 0 118↙，　　见图 12.13(c)。

? CS XW1 0 190↙，　　见图 12.13(d)。

每次运行，自动改变的是前进的步长，而转角不同。

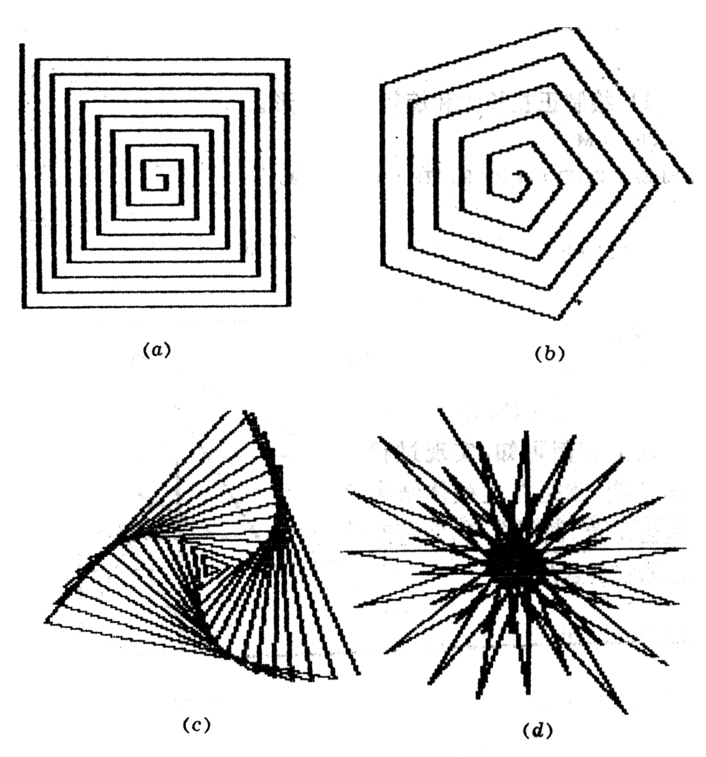


图 12.13　旋涡线的变化

例 13. 步长固定，自动改变转角的曲线图形。

```
TO XW2 :A :B
FD :A RT :B
XW2 :A :B+10
END
```

运行此过程：

? CS XW2　10　3↙

```
? CS XW2   10   5↙
? CS XW2   15   1↙
? CS XW2   15   2↙
```

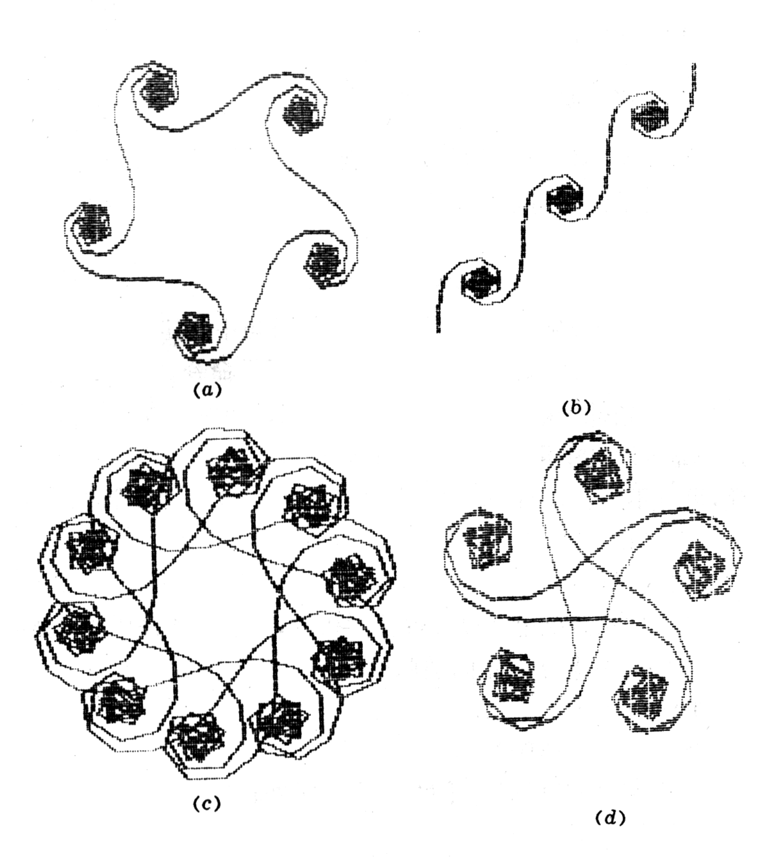


图 12.14　曲线图

运行结果见图 12.14(a)～(d)。注意：

①当图形超过边界时，会造成图形错乱，若准备打印该图形，要及时按下 **Ctrl＋G** 键，强行停止运行。

②每次重绘图形时，请用“CS”命令清屏幕，以后的示例中，不再书写“CS”命令。

12.4.3 递归过程的停止

除用 CTRK＋G 键强行停止递归过程的运行外，最好在过程体中用条件语句来自动控制递归过程的运行。

1. 条件语句

格式：**IF** 〈条件〉 **THEN** [〈命令序列〉]

作用：如果满足条件，那么就执行语句中指定的命令序列，如果不满足条件，则执行该条件语句的后继命令。

其中条件一般是由关系运算符(〉、〈、＝)所构成的关系表达式。命令序列中可以是多个命令或过程的集合，“THEN”为待选项，可以不写。(固化 LOGO 为 IF 〈条件〉 THEN 〈命令序列〉)。

2. 停止命令

格式：**STOP**

作用：此命令用来停止过程的执行，一般同条件语句联用。

例 14. 按条件停止的递归过程。

```
TO CF6 :M
IF :M>100 [STOP]
REPEAT 2[FD :M RT 90 FD :M/2 RT 90]
RT 30
CF6 :M+10
END
```

此过程同过程 CF5，仅增加了条件停止语句。

```
? CF6      30↙
```

可以看到当图形增加到指定大小(：M>100)时，过程会自动停止运行。

此过程的停止条件，也可以利用控制转动角度的大小来实现。

例 15. 用转角条件实现停止的递归过程。

```
TO CF7 :M :N
IF :N>270[STOP]
REPEAT 2[FD :M RT 90 FD :M/2 RT 90]
RT :N
CF7 :M+10 :N+30
END
? CF7 30 20↙
```

此过程运行时，当图形转到指定位置(：N>270)时，过程即会自动停止执行。注意：

①条件语句中的各部分必须写在同一行中，不能人为换行，一行写不下时，系统会自动换行。

②条件语句不能单独使用，必须置于程序之中。

12.5 过程的修改与编辑

12.5.1 定义过程时的出错修改

定义过程时出现的错误，有两种可能：

①定义中发现当前行出错，可用“◁”键，将光标退回到出错处，重新输入正确字符即可。这种方式，在光标退回时，将清除掉沿途的正确字符。所以还可以用 CTRL+B 控制键来实现光标退回，它可以保留光标经过的字符不被清除。当键入正确字符时，

保留字符自动右移。

②定义过程在换行后发现前面出错，或者在定义后运行时发现出错，则只有使系统进入编辑状态后方能进行修改。

12.5.2 在编辑状态下修改过程

1. 进入编辑状态

格式：**EDIT/ED** "〈过程名〉

(固化 LOGO 为 ED〈过程名〉)

作用：使过程处于编辑状态。

键入 ED "〈过程名〉，然后敲回车键，清屏幕并使给定过程的程序段显示在屏幕上，允许进行修改。注意：

使用"ED"命令时，过程名前必须加双引号""，而过程名后又不能带引号，这种格式在后面介绍的过程的调阅、清除和读写时都如此。(固化 LOGO 除读写外，均不加此双引号)。

2. 删除字符控制键

利用如下的光标控制键，可以删去过程中的字符。

Ctrl＋D：删除光标处字符，右端字符依次左移一格。

Ctrl＋K：删去该行光标所在处右端的全部字符(包括自动换行字符)。

◁键：左移光标，即删去光标左侧字符(固化 LOGO 为 ESC 键)。

3. 光标移动控制键

Ctrl＋P：光标上移一行。

Ctrl＋N：光标下移一行。

Ctrl＋A：光标移到所在行行首。

Ctrl＋E：光标移到所在行行尾。

Ctrl＋B：左移光标(不删除字符)。(固化 LOGO 为◁键)

▷键：右移光标(不删除字符)。

Ctrl＋O：将光标所在行下移，留出一个空行，光标留在空行行首。

空格键：光标连同光标处字符一起右移，相当于在行中插入空格。

回车键：将光标右侧字符连同光标一起，移到下行。

4. 修改或编辑

①插入新行

将光标移到待插入新行的行首位置(可用 Ctrl＋P 或 Ctrl＋N 与 Ctrl＋A 实现)，再用 Ctrl＋O 使该行下移而空出一个新行，键入待插入的字符行。

②删除一行

将光标移到待删行的行首，用 CTRL＋K 即将该行字符全部删去(包括自动换行部分)。

③插入空格

将光标移到需插入空格处右边一个字符上，敲空格键。

④插入新字符

将光标移至需插入新字符处的右边一个字符上，直接键入新字符，原来光标右侧字符自动右移。

⑤删除字符

将光标移至需删除字符处，敲 CTRL＋D 键，就可去掉该字符，光标右侧的字符自动递推补位。也可将光标移至需删除的字符右侧，敲◁键，同样可以删去该字符。

5. 退出编辑状态

修改、编辑完毕，敲 CTRL＋C，保存修改后退出编辑状态，屏幕显示“〈过程名〉 DEFINED”字样，重新出现提示符“?”，过程的编辑即告结束。

若修改、编辑完毕，欲撤消修改并维持原来过程内容的话，可敲 CTRL＋G 键退出，则屏幕上只出现提示符，说明修改无效。注意：

①在编辑状态下，也可以定义过程，由于这种状态下便于随时修改，实际定义过程中多用此法。

②不能修改过程名，若修改了过程名而不修改过程体时 则在退出编辑后，系统中将会出现同过程体而不同过程名的两个过程。

12.6 过程的调阅与清除

1. 过程调阅

①调阅过程名

格式：**POTS**

作用：屏幕用全文本显示方式，显示系统中全部的过程名。

②调阅单个过程内容

格式：**PRINTOUT/PO** "〈过程名〉

(固化 LOGO 为 PO〈过程名〉)

作用：屏幕显示该过程的全部内容(不能修改)。

③调阅全部过程内容

格式：**POALL**

(固化 LOGO 为 PO ALL)

作用：屏幕上依次显示系统中全部过程的内容。

在显示中，敲 Ctrl＋W 可暂停显示，再敲任意键，又可继续显示后继过程。

2. 过程清除

①清除指定过程

格式:**ERASE/ER** "〈过程名〉

(固化 LOGO 为 ER 〈过程名〉)

作用:将指定过程从系统中清除,清除后可用命令"POTS"观察。

②清除全部过程

格式:**ERALL**

(固化 LOGO 为 ER ALL)

作用:清除系统中的全部过程,使用时要慎重。此命令还可以用作"清内存"。

③清除部分过程

格式:**ER**[〈A 过程名〉〈B 过程名〉……] (固化 LOGO 无)

作用:清除方括号中指定的若干过程。

12.7 过程的存盘与读入

对于有用的过程,有必要用存盘命令把它存入软盘保留下来,以便经常使用。

1. 存盘命令

格式:**SAVE** "〈文件名〉

作用:将内存中的所有过程存入磁盘。

LOGO 的存盘特点是用任意一个文件名,就可以把系统内存中的全部过程一次存入,磁盘上记录的是文件名,该文件中包括了当前内存中的全部过程。文件名的取名同过程名取名一样。

注意:

①每次存盘时所取的文件名彼此不能相同,否则屏幕将显示出错信息,表示磁盘中已有该文件,不再接收。

②将已经存盘的过程调出使用时，若在内存中又建立了若干新的过程，则将新过程存盘时，会把旧过程再一次存入，从而多占了磁盘存储区。为此在新过程存盘时，应先清除内存中的旧过程。清除时，千万不要用 ERALL 命令，以免将新过程一起清掉，因为 ERALL 要清内存。建议采用如下方式处理：先用 POTS 调阅过程名，再用清除部分过程的命令（见 12.6-2-③），将旧过程删去。

2. 读盘命令

格式：**LOAD "〈文件名〉**

作用：将磁盘上指定的文件名所包括的全部过程读入内存，文件名必须是磁盘中已有的。当文件名不符时，将显示出错信息，为此系统提供了查阅磁盘文件目录的命令。

3. 列磁盘文件目录

格式：**CATALOG**

作用：在屏幕上显示出磁盘中所存的全部文件的名字。

例 16. 显示磁盘文件目录。

? CATALOG↙　　屏幕显示：

T　12　HDQ　.LOGO

T　16　HCQ　.LOGO

T　8　LX　.LOGO

其中“T”表示文本文件，后面的数字表示该文件在磁盘中占用的扇区数，数字之后是文件名，文件名后的“.LOGO”为文件的扩展名，表示该文件属 LOGO 文件。在存盘和读盘时，均不需键入“.LOGO”字符。

4. 删除磁盘文件

格式：**ERASEFILE "〈文件名〉**

作用：将磁盘中指定的文件删除。使用时，要慎重，避免删去

有用的文件。建议在删除某文件之前，最好用读盘命令先将该文件读入内存，然后再删除，若发现删除错了，还可以将读入内存中的文件重新写盘。

前面介绍的过程和图形的存取是针对 APPLE-LOGO 而言的，对于固化 LOGO，所用命令稍有不同。如果要在固化 LOGO 下进行过程和图形的存取，必须这样操作：

①启动方式

先用 DOS 启动中华学习机，使之处于“]”状态，再敲入：

]MAXFILE 1↙

]LG↙

这样，固化 LOGO 就有存取功能了。

②存取命令格式

SAVE "〈过程文件名〉——过程存盘

READ "〈过程文件名〉——将过程读入内存

SAVEPICT "〈图形文件名〉——图形存盘（APPLE-LOGO 无此命令）

READPICT "〈图形文件名〉——读入图形（APPLE-LOGO 无此命令）

CATALOG——列磁盘目录

ERASEFILE "〈文件名〉——删除磁盘文件

ERASEPICT"〈图形文件名〉——删除磁盘图形文件（APPLE-LOGO 无此命令）

12.8 过程与图形的打印

有时需要将程序、结果或图形打印在宽行纸上，这就要用到打印命令。

1. 联机/脱机命令

格式:. **PRINTER** 〈1/0〉

（固化 LOGO 为 OUTDEV〈1/0〉）

作用:参数为"1",则接通打印机,使之处于等待状态。参数为"0",则断开打印机。使用时注意此命令前的点号"·"。

2. 过程打印命令

①打印全部过程内容

格式:**POALL**

作用:打印出内存中的全部过程内容。

在联通打印机后,屏幕上的提示符消失,仅剩下光标,此时敲入 POALL↙,就能自动启动打印机工作,从而打印出内存中的全部过程。打印时,屏幕上的光标消失,待打印结束方重新出现。

②打印单个过程内容

格式:**PO** "〈过程名〉

作用:打印内存中已有的指定过程的内容。

③打印过程名字

格式:**POTS**

作用:只打印内存中全部的过程名字。

以上三条过程打印命令与过程的调阅命令对应相同,区别仅在于是否已联通打印机。

3. 图形打印命令

格式:**CHAR** 17

作用:打印屏幕上显示的全部图形(相当于屏幕拷贝)。

打印图形时,由于纵横比例的影响,会造成某些图形变形,此时应适当修订屏幕纵横比例参数。

4. 打印运行结果

格式:**PR**(〈过程名〉〈实在参数 1〉〈实在参数 2〉……)

作用:打印运行过程中输出的数值和字符。

5. 打印格式的设置

格式:. **DEPOSIT** 1913 〈**X**〉

作用:当 X=
- 1 正常打印
- 33 反相打印
- 65 正常放大打印
- 97 反相放大打印

开机后,若不使用此命令,则打印机自动按正常打印方式工作。

12.9 彩色绘图

1. 屏幕底色的设置

格式:**SETBG** 〈**X**〉　　　　(固化 **LOGO** 为 **BG** 〈**X**〉)

作用:设置屏幕颜色。

其中 X 为颜色代码,其值为 0~5 整数,与 X 值对应的颜色如下:

X 值:	0	1	2	3	4	5
颜色:	黑	白	绿	紫	黄	兰

2. 画笔颜色的设置

格式:**SETPC** 〈**X**〉　　　　(固化 **LOGO** 为 **PC** 〈**X**〉)

作用:设置画笔颜色。

X 值范围及其对应的颜色同上。

3. 屏幕底色与画笔颜色的配置

当屏幕底色与画笔颜色的配置不同时,彩色绘图的效果是不同的。若配置不当,甚至无法在屏幕上显示图形。配置的关系

见表 12.1：

注意：

①当屏幕底色与画笔颜色相同或在某些特殊配置时，屏幕上显示不出图形，在上表中用“×”表示。

②开机后，自动进入彩色绘图状态，屏幕底色设为黑色，画笔设为白色。

③当使用单色显示器或黑白电视机时，无彩色显示，但可以观察到灰度变化。

表 12.1　屏幕底色与画笔颜色配置表

画笔 \ 效果 \ 屏幕	0	1	2	3	4	5
0	×	0	0	0	0	0
1	1	×	1	1	1	1
2	2	2	×	2	×	4
3	3	3	3	×	5	×
4	4	4	×	2	×	4
5	5	5	3	×	5	×

例 17. 变底色过程。

```
TO DS :X
IF :X>5 [STOP]
SETBG :X
WAIT 50
DS :X+1
END
? DS 0 ↙（观察运行中屏幕底色的变化）
```

改变画笔颜色：

```
TO BS :X
IF :X>5 [STOP]
```

```
SETBG 0 SETPC :X
FD 30 RT 60
WAIT 50
BS :X+1
END
? BS 0 ↙
```

观察绘图中各线段颜色的变化，并想一想为什么第一段线段没有显示？

12.10 图形的擦抹

在屏幕上绘制的若干图形中，发现有的图形是错误的，怎么办？是全部清除再从头绘制，还是只将错误的图形单独擦掉？显然应该选择后者。下面介绍图形擦抹的方法。

我们已经知道，当画笔颜色与屏幕底色相同时，画笔绘图就不能显现出图形，若使画笔再次经过已绘好的图形时，还会把其所到之处的图形擦掉。基于上述现象，我们只要在画笔与屏幕底色相同时，使画笔再执行一次原过程，就可以达到把原图擦掉的目的。

例 18. 观察擦图现象。

试运行过程“FC1”（见 12.3 例 5）。

? FC1 ↙ （屏幕显示一风车图形）

现欲擦掉此图，请键入：

? PU HOME PD↙

? SETPC 0↙

? FC1 ↙ （观察图形的擦抹）

注意：

①上述过程运行时对应的屏幕底色代码 X=0，否则要调整命令“SETPC”后面的参数 X 的值。

②擦图完毕，要及时改变画笔颜色，以备运行其它过程，否则再画的图形同样显现不出来。

利用上述方法，还可以使所绘图形在运行中出现“闪烁”或“动态”的效果。

例 19. 闪烁的圆点。

```
TO SS
HT SETPC 1 DOT [0 0]
WAIT 20 SETPC 0 DOT [0 0]
SS
END
```

运行此过程，可以看到在坐标原点处，有一个闪烁的亮点。

例 20. 移动的小球。

```
TO DQ
SETPC 1 RT 45
REPEAT 4[FD 2 LT 90]
WAIT 20 SETPC 0
REPEAT 4[FD 2 LT 90]
LT 45 FD 5
DQ
END
```

运行此过程，可以看到从坐标原点处开始，有一个不断向上移动的小圆球。

前面几个过程用到一条新命令：WAIT 〈X〉，此命令的作用是延时，延时的长短由 X 的值大小决定。（固化 LOGO 中无此命令，当需要延时时，可以用 REPEAT 〈X〉[ ]代替。）

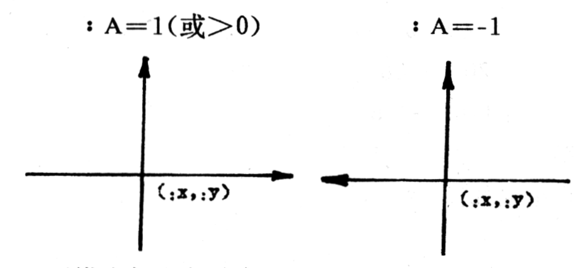
12.11 绘图程序选编

12.11.1 直角坐标系

例 21. 画直角坐标轴。

```
TO ZJ :X :Y :A
HT PU SETPOS SE :X :Y PD
SETY 115 SETPOS SE :X-4 95
PU SETX :X+4 PD
SETPOS SE :X 115
BK 230 SETY :Y SETX 95
IF :A=-1 [SETX -135 SETPOS SE -115 :Y+4 PU BK 8 PD
          SETPOS SE -135 :Y STOP]
SETPOS SE 75 :Y+4 PU BK 8 PD
SETPOS SE 95 :Y
SETX -135 SETX :X
END
```

说明：:X，:Y 为坐标系原点坐标，:A 为横坐标方向代码，如下图：



例 22. 画横坐标和纵坐标。

①横坐标

```
TO HB :H :Y :A :N
PU SETY :Y PD
IF :N=-1 [PU HOME PD STOP]
SETX :H+(70-:H)*:A*:N
FD 2 BK 2
HB :H :Y :A :N-1
END
```

说明：:H 为刻度线终点横坐标，:Y 为刻度线所在纵坐标，:N 为刻度数，:A=1/:N。

②纵坐标

```
TO ZB :X :Z :A :N
PU SETX :X PD
IF :N=-1 [PU HOME PD STOP]
SETY :Z+(90-:Z)*:A*:N
SETX :X+2 SETX :X
ZB :X :Z :A :N-1
END
```

说明：:Z 为刻度线终点纵坐标，:X 为刻度线所在横坐标。

运行上述过程：

```
? ZJ -20 0 1
? HB -110 0 1/12 12
? ZB -20 -60 1/10 10
? ZJ -110 -75 1
? ZJ 70 75 -1
```

运行结果如图 12.15

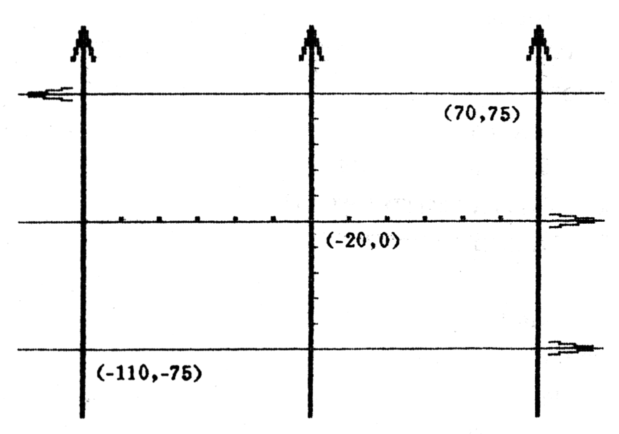


图 12.15 横坐标和纵坐标

说明：在使用"HB"和"ZB"过程时，应考虑坐标轴的位置，使刻度之一落在轴上。

12.11.2 画字母

例 23. 绘出字母"LOGO"。

①"L"

```
TO L :X :Y
PU SETPOS SE :X :Y PD
BK 60 RT 90 FD 40
PU HOME PD
END
```

②"G"

```
TO G :X :Y
PU STPOS SE :X :Y PD
LT 60 REPEAT 24[FD 5 LT 10]
LT 60 BK 10 FD 30 LT 90
FD 15 BK 30
```

```
PU HOME PD
END
```

③"O"

```
TO O :X :Y
PU SETPOS SE :X :Y PD
RT 90 REPEAT 36[FD 5 RT 10]
PU HOME PD
END
```

④"LOGO"

```
TO LG
L -120 50 O -40 50
G 50 45 O 100 50
END
```

运行 LG 过程：? LG ↙，运行结果如图 12.16。

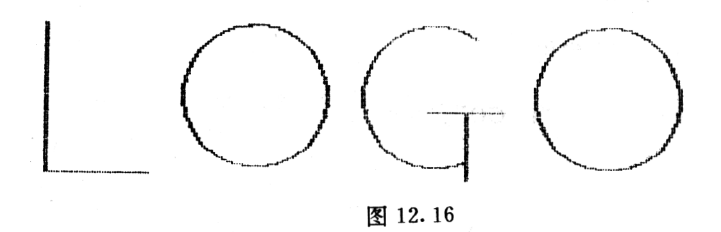


图 12.16

12.11.3 折线图形

例 24. 画单元折线。

```
TO DZ
REPEAT 2[FD 40 RT 90]
REPEAT 2[FD 20 RT 90]
FD 40 RT 90
REPEAT 2[FD 10 RT 90] FD 20
END
```

运行此过程,屏幕显示如图 12.17(a)所示。

例 25. 画折线图形。

①折线图形 1

```
TO T1
CS REPEAT 4[DZ]
END
```

②折线图形 2

```
TO T2
CS REPEAT 8[DZ LT 45 FD 70]
END
```

③折线图形 3

```
TO T3
CS REPEAT 4[DZ DZ LT 90]
END
```

运行上述三过程,屏幕显示如图 12.17(b)～(d)。

12.11.4 圆弧图形

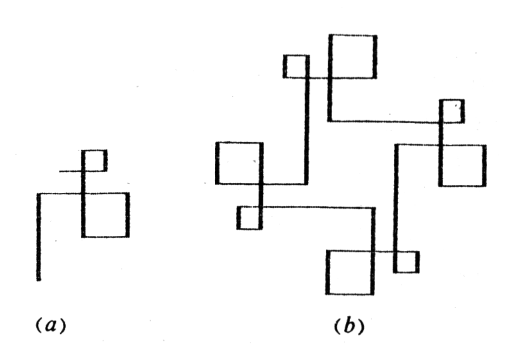
例 26. 画圆弧。

```
TO YH :N :R :A
REPEAT :N[FD :R*3.14/18 RT 5* :A]
END
```

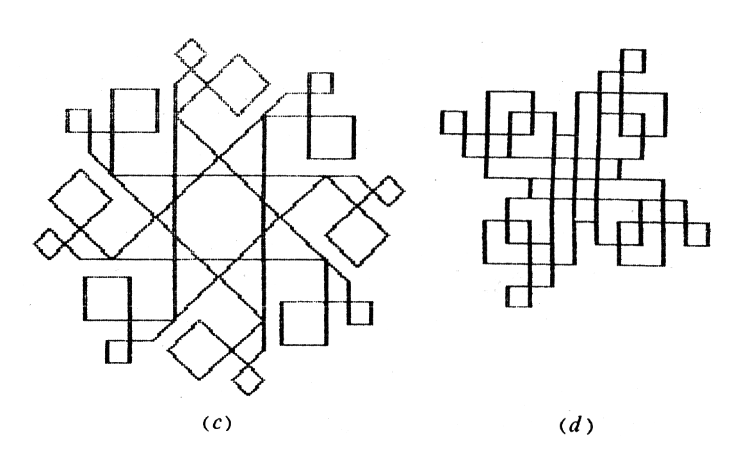
说明:圆弧相当于边数较多的正多边形的一部分。:N 为边数,当 :N=72 时,为整圆;:A 为左右圆弧代码,:A=1 为右圆弧,:A=-1 为左圆弧;:R 为对应圆半径。

例 27. 用圆弧构成图形。

①对圆



(*a*)　　　　　　　　(*b*)



(*c*)　　　　　　　　(*d*)

图 12.17　折线图

```
TO DY :R :N
IF :N<1 [STOP]
YH 72 :R 1 YH 72 :R -1
DY :R- :R/ :N :N-1
END
```

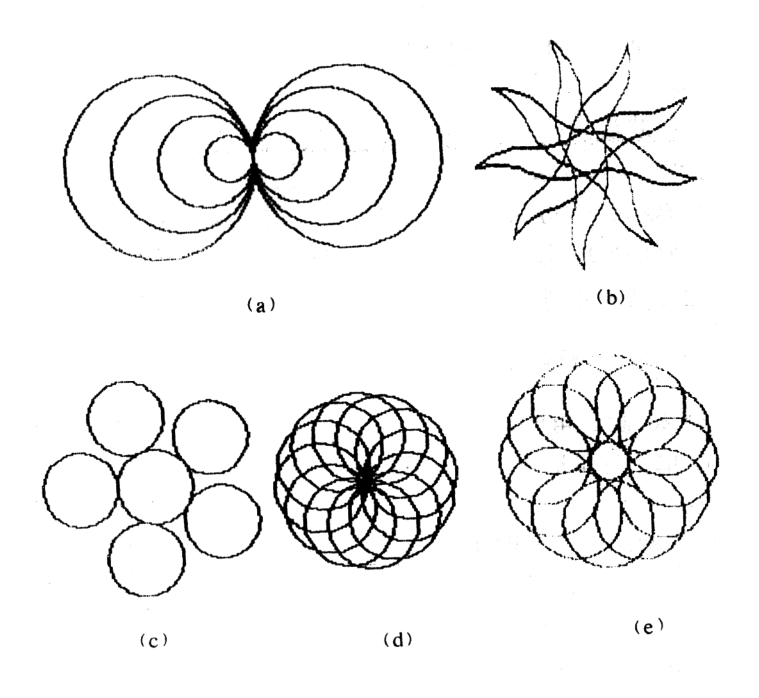


图 12.18 圆弧图形

运行此过程：

? DY 25 4↙　　结果如图 12.18(a)。

②火球

```
TO HQ
REPEAT 9[REPEAT 2[YH 9 20 1 YH 9 20 —1] RT 160]
END
```

运行此过程：

？ HQ↙，结果如图 12.18(b)。

③花球

```
TO YQ :M :R1 :R2
REPEAT :M[YH 72 :R1 -1 YH 72/ :M :R2 1]
END
```

运行此过程：

```
? YQ 5 10 10↙
? YQ 12 15 0↙
? YQ 12 15 5↙
```

运行结果分别为图 12.18(c)～(e)。

12.11.5 三角图形

例 28. 画三角形。

①单个三角形

```
TO SJ :L
REPEAT 3[FD :L RT 120]
END
```

②旋转三角形

```
TO XS1 :L :N :A
IF :N<1 [STOP]
SJ :L RT 360/ :A
XS1 :L :N-1 :A
END


TO XS2 :N :L :A
REPEAT :N [FD :A LT 30 SJ :L RT 30 BK :A RT 360/ :N]
END
```

运行上述二过程：

```
? XS1 45 10 10↙
? XS2  25  25↙
```

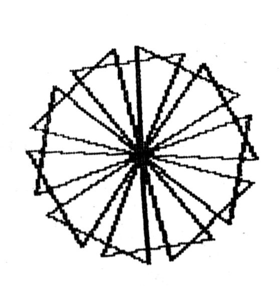
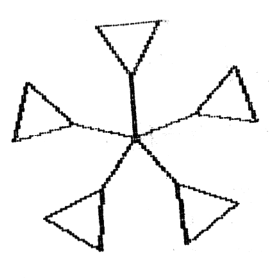



图 12.19

结果分别如图 12.19(a),(b)。

例 29. 蝴蝶。

①画翅膀

```
TO CB :L
IF :L<20 [STOP]
SJ :L LT 180 SJ :L
CB :L-20
END
```

②画触角

```
TO CJ :A
LT 15 * :A FD 40 LT 120 * :A FD 15
BK 15 RT 120 * :A BK 40 RT 15 * :A
END
```

说明::A 为左右触角代码::A=1 为左,:A=-1 为右。

③画蝴蝶

```
TO HD :L
RT 60 CB :L RT 90 SJ 40
```

```
LT 150 CJ 1 CJ -1
END
```

运行此过程：

? HD　80　↙，结果如图 12.20。

12.11.6　荷花

例 30. 水中荷花。

①画花瓣

```
TO H :X :L
PU SETX :X PD
REPEAT 2[REPEAT 9[FD :L RT 10] RT 90]
REPEAT 2[REPEAT 9[FD :L LT 10] LT 90]
END
```

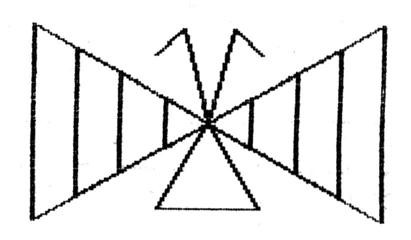


图 12.20　蝴蝶图形

②画荷花

```
TO HH :R :X :L :A
RT :R
REPEAT 2[H :X :L RT 45]
LT 135 H :X :L RT 45
REPEAT 4[BK 10 RT :A]
PU HOME PD
END
```

③画水面

```
TO SM
PU SETPOS [-110 -20] PD RT 180
REPEAT 19[REPEAT 18[FD 1 LT 10] RT 179.5]
PU HOME PD
END
```

④画荷花池

```
TO HC
HT HH 0 0 6 4 HH 20 -80 4 -10
HH -20 75 3 10 SM PU HOME PD
END
```

运行此过程：

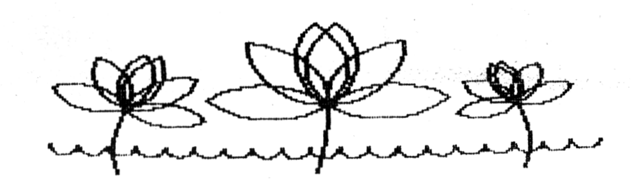


图 12.21　水中荷花

? HC↙，结果如图 12.21。

12.11.7　树林

例 31. 独木不成林。

①树枝

```
TO SC :N :L :A
IF :N=0 [STOP]
LT 30* :A FD :L*2 SC :N-1 :L :A
BK :L*2 RT 60* :A FD :L
SC :N-1 :L :A BK :L LT 30* :A
```

```
END
```

说明：：A 为树倾斜方向代码，：A>0 左斜，：A<0 右斜，：A 不能为 0。

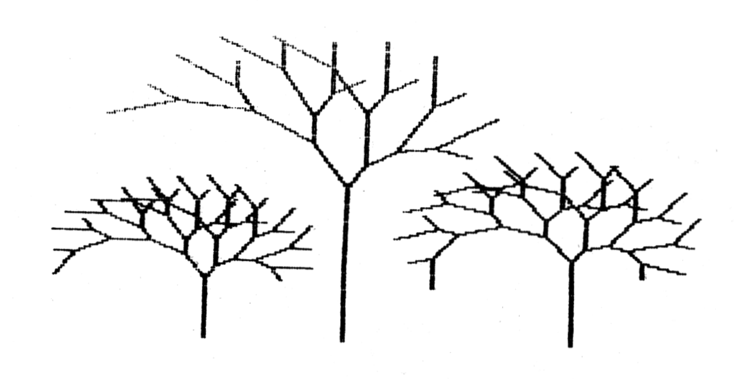


图 12.22

②树

```
TO S :X :Y :N :L :A
PU SETPOS SE :X :Y PD
SC :N :L :A SETY -60
END
```

③树林

```
TO SL
S -75 -25 5 8 1
S -20 30 4 15 .9
S 70 -10 5 9 1.2
END
```

运行此过程：

？SL↙，结果如图 12.22。

12.11.8 闪光的红星

例 32. 红星照我去战斗。

①五角星

```
TO WX :L
MAKE "A :L * (SIN 36)/SIN 126
MAKE "B :L * (SIN 18)/SIN 126
REPEAT 5 [FD :L LT 162 FD :A LT 54 FD :B LT 108 FD :B LT
          54 FD :A RT 18 BK :L RT 72]
END
```

说明：MAKE 为赋值语句。

②闪光

```
TO SG :L
REPEAT 30 [PU FD 1.2 * :L PD FD .4 * :L PU BK 1.6 * :L PD
           RT 12]
END
```

③闪闪的红星

```
TO SX :X :Y :L
HT PU SETPOS SE :X :Y PD
WX :L SG :L
END
```

说明：:X，:Y 为五角星中心位置坐标。运行此过程：

？SX －50 10 50↙，结果如图 12.23。

12.11.9 五星红旗

例 33. 飘扬吧！五星红旗。

①旗

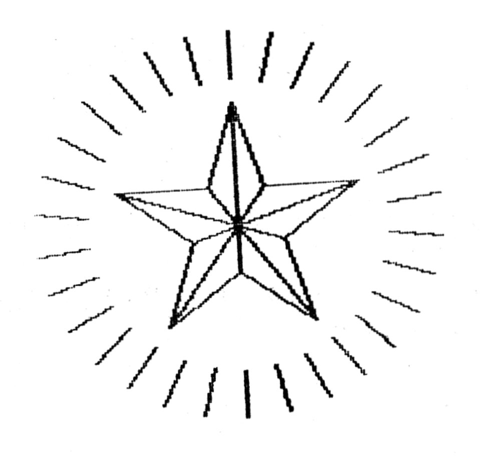


图 12.23　闪闪的红星

```
TO Q
PU SETPOS [－120 －50]
PD
REPEAT 2[FD 130 RT 90
FD 240 RT 90]
BK 20 PU HOME PD
END
```

②星

```
TO X :X :Y :L
PU SETPOS SE :X :Y PD
WX :L
END
```

③五星红旗

```
TO GQ
HT Q X －80 45 25
X －40 60 10 X －33 33 10
X －50 10 10 X －80 5 10
END
```

运行此过程：

? GQ↙，显示如图 12.24。

12.11.10　**时钟**

例 34. 时间就是生命。

①刻度盘

```
TO KD
REPEAT 12 [PU FD 60 PD FD 10 PU BK 70 PD RT 6 REPEAT 4[PU FD
          65 PD FD 5 PU BK 70 PD RT 6]]
END
```

②指针

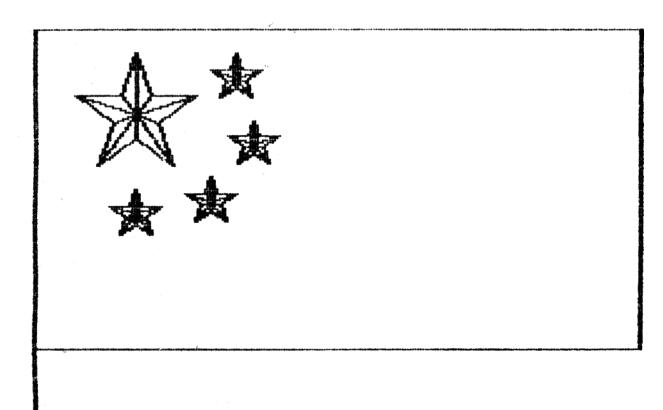


图 12.24

```
TO ZZ :T :S :H
SETPC 1 SETH :S FD 55 BK 55
SETPC 1 SETH :H FD 35 BK 35
WAIT :T
SETPC 0 SETH :S FD 55 BK 55
SETPC 0 SETH :H FD 35 BK 35
ZZ :T :S+6 :H+.5
END
```

说明：S 为分针位置，：H 为时针位置，位置以顺时偏离 Y 轴角度计，初值为 0。

③时钟

```
TO SZ :T :S :H
HT KD ZZ :T :S :H
END
```

运行此过程： SZ 10 0 0 ↙

屏幕显示一个从 12:00 开始走动的时钟。图 12.25 中所示

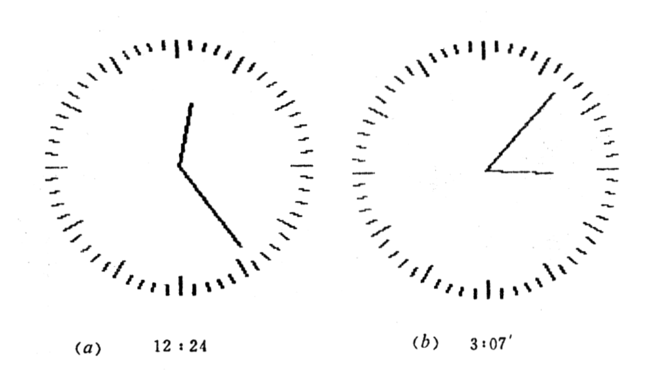


图 12.25

为任意二个时刻钟的状态。

说明:欲使此过程暂停运行(即指针停走),可敲 CTRL+Z 键,欲恢复运行,键入 CO↙。也可以敲 CTRL+W 键暂停,敲任意键恢复。

12.11.11 指定键的绘图程序

```
TO LJ
ST MAKE "A RC
IF :A="Q [CS LJ]                        (清屏)
IF :A="I [FD 10 LJ]                     (前进 10 步)
IF :A="M [BK 10 LJ]                     (后退 10 步)
IF :A="J [LT 30 LJ]                     (左转 30 度)
IF :A="K [RT 30 LJ]                     (右转 30 度)
IF :A="U [PU LJ]                        (抬笔)
IF :A="O [PD LJ]                        (落笔)
IF :A="1 [SETBG 1 SETPC 0 LJ]           (白底黑字)
IF :A="0 [SETBG 0 SETPC 1 LJ]           (黑底白字)
```

END 说明：

①右方注明的绘图状态，均可由敲对应的单键完成（如：敲"I"键，海龟即前进 10 步）。

②每次只能敲一个指定键，并且要在海龟状态变换后，方可再敲键。

③敲其它任意键，即可退出此过程。

④此过程还可随意扩充。

⑤过程中的 RC 命令，作用为等待输入一个字符，相当于 BASIC 中的 GET 语句。

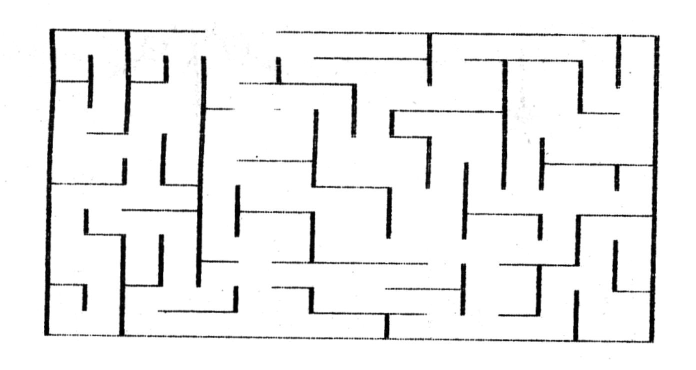


图 12.26 迷宫图案

图 12.26 所示的"迷宫"图案，即由反复使用"I"、"M"、"J"、"K"、"U"、"O"等键获得。

习题十二

1. 请用画字的方法分别编一个写“X”和写“Y”字符的过程。

2. 请将纵横比例设置命令，联接打印机命令，打印图形命令和脱开打印机等常用命令，分别编为过程，以便随时调用。

3. 编写绘制下列图形的程序，并运行之。

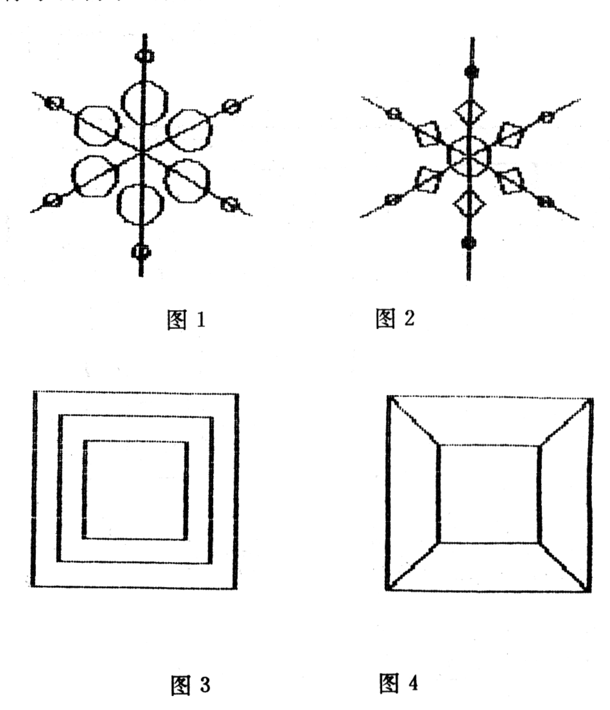


图 1　　图 2

图 3　　图 4

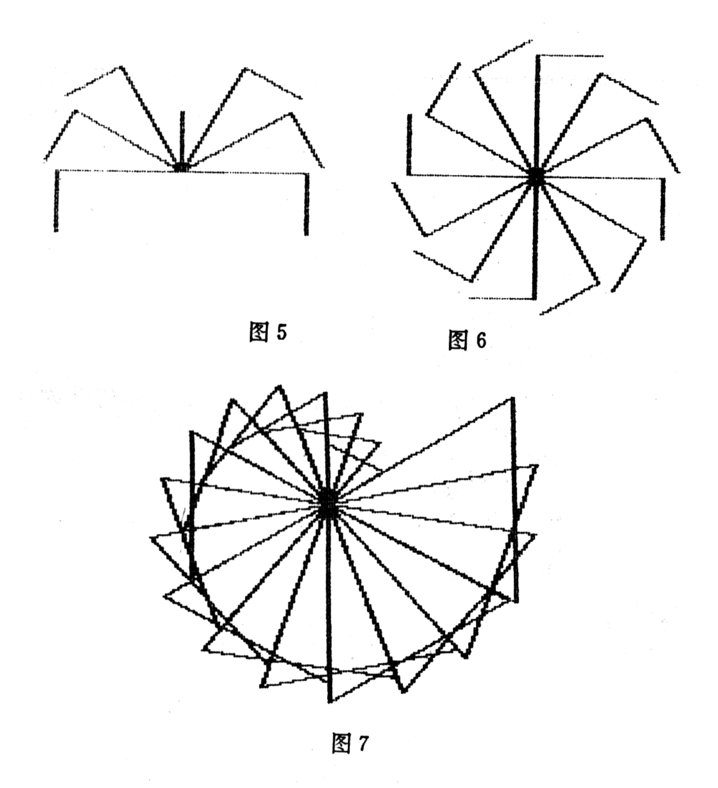


图 5　　图 6

图 7

第十三章　数、字和表处理

13.1　LOGO 的数和算术运算

1. LOGO 的数

①整数,即不带小数点的数。

如:1989,13,—60,285。

整数在 LOGO 中的使用范围为 $-2^{31} \sim 2^{31}$,计算精度为 9 位有效数字。

②实数,即带小数点的数。

如:4.73,—13.85,3.05,—28.0。

实数在 LOGO 中的使用范围为 $10^{-38} \sim 10^{38}$,计算精度为 7 位有效数字。

2. 基本算术运算

①基本运算符

+(加)、—(减)、*(乘)、/(除)

②运算规则

有括号时则括号优先,无括号时则先乘除,后加减,这与数学中的运算规则相同。如:

100—16×4+28÷7　　写成 LOGO 表达式为:

```
100 —16 * 4 +28 /7
     ↑     ↑   ↑     ↑
     ③     ①   ④     ②
```

运算顺序依次从①～④。注意：

①LOGO 运算中，只能使用圆括号。

②当除式为表达式或负数时，必须加圆括号，如：

60÷(18—6)，应写成：60/(18—6)

60÷(-5)，应写成：60/(—5)

③当进行算术运算时，其中只要有一个数为实数，其运算结果必为实数。

3. 数的科学记数法

当 LOGO 中表示的数太大或太小时，系统会自动按科学记数法表示，用户也可以用科学记数法输入数据。

科学记数法表示的数据格式为：

〈数〉〈E 或 N〉〈指数〉

其中：〈数〉是只有一位整数的实数，当然可以全是整数，也可以是纯小数，其值可正可负。〈指数〉必须为正整数，不能是实数，也不能是负数，指数本身的正负由 E 或 N 表示。“E”表示的是 10 的正整数次方，“N”表示的是 10 的负整数次方。如：

10325⇒1.0325×10^4 可以写成：1.0325E4

0.010325⇒1.0325×10^{-2} 可以写成：1.0325N2

以下的写法是错误的：

3E5.1　　　(指数不能用实数)

2.7E—4　　 (指数不能用负数)

9.3　E2　　(数与符号间不能有空格)

13.2　变量与函数

13.2.1 变量

变量即前面程序中用到的形式参数，它可以用在数值和函数运算中。变量名可以用一个或多个字母和数字符号组成。在为变量取名时，必须注意 LOGO 语言的命令不能再用作变量名。含有多个字符的变量名中不能夹杂空格或运算符号。

以下的变量名是正确的：

：X，：Y，：AM，：A2，：M123，：3B

以下的变量名是错误的：

：FD，：CS，：A＋B，：M　N，：4 注意：

变量名前一定带有冒号，这是 LOGO 变量同其它语言变量最明显的区别。

13.2.2 基本函数

1. 正弦函数　SIN：X

求 X 的正弦值。注意：

①SIN 与其后的：X 之间必须用空格隔开（其它函数亦如此）。

②：X 的取值为角度值，当其值大于 360 度时，会自动按减去 360 度的整倍数后的余数计。

③当：X 的值为弧度时，必须换成角度值再进行运算，换算公式为：角度值＝弧度值×180/3.14

例 1.

？PR　SIN　405↙　　　（相当于求 SIN（405－360））。

　　.707107　　　　　　　（结果）

2. 余弦函数　COS：X

求 X 的余弦值。

例 2.

? PR　COS　145↙

　　-0.819152　　　　　　　　　(结果)

3. 平方根函数　SQRT　:X

求 X 的平方根。注意:X 值不能小于零。

例 3.

? PR　SQRT　100↙

　　10　　　　　　　　　(结果)

4. 随机函数　RANDOM　:X

随机产生从 0～(X－1)之间的任意正整数,X 值必须为正整数。

例 4.

? PR　RANDOM　10↙

　　4　　　　　　　　　(结果)

5. 四舍五入取整函数　ROUND　:X

将 X 值四舍五入取整。

例 5.

? PR　ROUND　6.8↙

　　7　　　　　　　　　(结果)

6. 求整商函数　QUOTIENT　:X　:Y

将 X/Y 的商取整(小数部分全舍),Y 值不能为零。

例 6.

? PR　QUOTIENT　10　3↙

　　3　　　　　　　　　(结果)

7. 求余数函数　REMAINDER　:X　:Y

求 X/Y 的余数,Y 值不能为零。

例 7.

? PR　REMAINDER　10　3↙

　　1　　　　　　　　　　　(结果)

8. 求和函数　SUM　:X　:Y

求 X+Y 的和。(固化 LOGO 无)

例 8.

? PR　SUM　3　5↙

　　8　　　　　　　　　　　(结果)

9. 求积函数　PRODUCT　:X　:Y

求 X×Y 的积。(固化 LOGO 无)

例 9.

? PR　RPODUCT　2　3↙

　　6　　　　　　　　　　　(结果)

注意:

当求和函数和求积函数中的参数为三个或三个以上时,需用圆括号将函数和参数括起来。

10. 反正切函数　ARCTAN　:C 或 :A/:B

其中参数 :C= :A/:B 为角 α 的正切值,求 α 值

例 10.

? PR　ARCTAN　(SQRT 3)/3↙

29.9995　　　　　　　(结果)

? PR　ARCTAN 1.732↙

60.002　　　　　　　(结果)

　　(固化 LOGO 为 ATAN　:A　:B)

11. 舍小数取整函数　INT　:A

:A 为实数,舍去全部小数,输出整数。

例 11.

? PR　INT　6.8↙

6　　　　　　　　　　　　（结果）

（固化 LOGO 为 INTEGER ：A）

以上基本函数可以和算术运算结合起来，组成运算表达式，函数之间也可以嵌套或者相互调用。函数中使用的变量，可以是表达式。

除基本函数外，也可以自行定义一些函数，作为用户自定义命令使用，见 13.4 节。

13.2.3　运算符的优先级

1. 中缀运算符

指算术运算符（＋、－、×、/）和比较运算符（>，＝，<），这些符号的特点是符号位于运算参数中间。

2. 前缀运算符

指基本函数（包括自定义函数），这些函数的符号位于运算参数之前。

3. 运算符的优先级

①中缀运算符优先于前缀运算符；

②算术运算符优先于比较运算符；

③优先级相同的按顺序执行，其中算术运算符中，乘除优先于加减。

④圆括号内的运算最优先，多括号时按先内后外的顺序运算。

例 10. 求$\sqrt{9}-\sqrt{4}$的值

若运算表达式写成：

? PR　SQRT9－SQRT4↙

2.64575

这个结果是错误的，这是因为此表达式运行时先作的是"9－SQRT4"，然后再对这个值求平方根所造成的。正确的写法应该是：

? PR　(SQRT9)－SQRT4↙

1

例 11. 求 SIN45°/COS45°＋1 的值

若运算表达式写成：

? PR　SIN45/COS45＋1↙

.904655

这个结果也是错误的，这是因为此表达式运行时先作的是"45/COS45＋1＝45/COS46"，然后再对其值求正弦所造成的。正确写法是：

? PR　(SIN45)/(COS45)＋1↙

2

13.3　表达式

13.3.1　运算表达式

由算术运算符及圆括号将数、变量及函数等按要求联接而成的式子，称为算术表达式。

如：:B * :B－4 * :A * :C

(537＋SQRT　100)/ :A

((SIN :X)＋ :Y)/COS :X＋15

等，均为正确地算术运算表达式，在书写表达式时，要注意以下几个问题：

①考虑运算优先级，根据要求，在表达式中适当地成对使用

圆括号。

②变量名前必须有冒号“:”,冒号与变量名之间不能有空格。

③变量名与函数名之间至少必须有一个空格,当变量与变量,变量与数及变量与函数之间有运算符或圆括号隔开时,可以不用空格。

13.3.2 条件表达式

1.单一条件表达式

格式:**IF**〈条件〉**THEN** [〈命令序列〉]

条件满足时,执行指定的命令序列。

2.“与”条件表达式

格式:**IF AND** (条件 1) (条件 2) [〈命令系列〉]

(固化 LOGO 的表达式为:IF ALLOF (条件 1) (条件 2)
〈命令〉)

这种条件表达式的意思是:当条件 1“与”条件 2 同时被满足时,才去执行指定的命令序列。而只要有一个条件不满足,则执行该表达式的后继命令。这种条件关系一般称“与”逻辑。

例 12.

IF AND (: L>100) (: Q>360) [STOP]

这个表达式表示:当 : L>100 与:Q>360 的条件同时满足时,才会执行停止命令。

注意:当条件为三个或三个以上时,必须用圆括号将“AND”和所有条件括在一起。如:

IF (AND (: L>100) (: Q>360) (: N<0)) [STOP]

3.“或”条件表达式

格式:**IF OR**(条件 1) (条件 2) [〈命令序列〉]

(固化 LOGO 的表达式为:IF ANYOF(条件 1)(条件 2)〈命令〉)

这种条件表达式的意思是:当条件 1“或”条件 2 中有一个条件满足,就去执行指定的命令序列。这种条件关系一般称“或”逻辑。

例 13.

IF OR (:X=0)(:X=180)[PR :Y STOP]

这个表达式表示:当 :X=0 或 :X=180 时,过程都会去执行打印 :Y 的值,然后停止。

同样,当条件为三个或三个以上时,必须用圆括号将“OR”和所有的条件括在一起。

4.“非”条件表达式

格式:**IF NOT** (条件)[〈命令序列〉]

(固化 LOGO 的表达式为:IF NOT〈条件〉〈命令序列〉)

表达式的意思是:当不满足该条件时,执行指定的命令序列。这种条件关系一般称为“非”逻辑。

例 14.

IF NOT (:A=0)[PR :B/:A STOP]

此式表示:当 :A 不等于 0 时,打印 :B/:A 的值,然后停止。

5.条件表达式的嵌套。

同重复命令和过程嵌套相似,条件表达式也可以用嵌套的办法来简化过程的编制。

如在某过程中有这样二个条件表达式,其后的命令序列完全相同:

IF AND (:A>0)(:B=0)[PR :X]

IF AND (:B>0)(:C<0)[PR :X]

不难看出，这两个条件语句之间又存在着“或”的关系，故可改写为：

IF OR (AND (：A＞0) (：B＝0)) (AND (：B＞0)
(：C＜0)) [PR ：X]，注意：

条件表达式一般不能放在递归调用语句(如果有递归调用语句的话)的后面，请读者想想看，这是为什么？

13.3.3 赋值与输出

1. 赋值命令

格式：MAKE ″〈变量名〉〈表达式〉

作用：将表达式的值赋给指定变量。

例 15.

MAKE ″A SQRT 200

将表达式“SQRT 200”的值赋给变量：A。注意：

①变量赋值，不必区分变量的类型，同一个变量可以赋于不同类型的数据(数、字符及表达式)。

②被赋值的变量名前加双引号“″”，后面不加。变量一旦被赋值，在调用时，必须在其前面加上冒号(可以把冒号看成是构成变量的一部分)。

2. 打印输出命令

①换行打印

格式：PRINT/PR〈表达式或变量〉

作用：在屏幕上显示表达式或变量的值。

例 16. 打印正弦函数值

TO ZXHS ：T
MAKE ″A SIN ：T
PR ：A

```
ZXHS  :T+15
END
```

运行此过程：

```
? ZXHS  15↙
.258819                    (以下为输出结果)
.5
.707107
.866025
   ⋮
```

②紧凑打印

格式:TYPE 〈表达式或变量〉

作用:在一行中(包括自动换行)按紧凑格式打印多个数据,数据间无空格。

如例 16 中,将 ZXHS 过程中的 PR :A 改为 TYPE :A,然后再运行:

```
? ZXHS  15↙
.258819.5.707107.866025……
```

(固化 LOGO 命令为 PRINT1〈参数〉)

③提示打印

格式:(PR〈字或表〉〈表达式或变量〉)

作用:用字或表作提示,与数据打印在同一行上。

如例 16 中,将 ZXHS 过程中的 PR :A 改为(PR[SIN] :T [=] :A),然后再运行:

```
? ZXHS  15↙
SIN  15=.258819
SIN  30=.5
SIN  45=.707107
```

SIN　60＝.866025

⋮

3.程序传送命令

格式:**OUTPUT/OP** 〈表达式〉

作用:将表达式的值传送给其它过程。

数值以及函数运算也可以用过程实现,当被其它过程调用时,其值或运算结果必须使用传送命令,才能使对方接收,其结果并不显示在屏幕上。

例 17.

```
TO M :X
PR :X* :X
END
```

试运行此过程:

```
? M 12↙
144                        (输出结果)
```

现将"M"过程嵌入"N"过程:

```
TO N :X :B
MAKE "A M :X
PR :A+ :B
END
```

再运行:? N 12 15↙,屏幕上提示出错信息,告诉使用者,在过程中调用了没有输出的过程。

现将过程"M"中的打印命令"PR",用传送命令"OP"代替,即:

```
TO M :X
OP :X* :X
END
```

然后再运行过程"N":

? N 12 15↙

159　　　　　　　　　　　　(输出结果)

由此可见,"OP"命令的作用是将某过程的结果传送给另一个过程作为该过程中的实在参数使用,所以在数值或函数运算中,当调用某过程时,一定要使用传送命令"OP"。

4. 文本清屏命令

格式:CLEARTEXT

作用:清除屏幕字符。

13.4　数值计算程序选编

例 18. 正切函数。

```
① TO TG1
   IF OF ( : X=90) ( : X=270) [PR[/] STOP]
   MAKE "Z (SIN  : X) /COS  : X
   PR  : Z
   END
```

过程建立之后,可以直接作为函数使用。

```
? TG1 45↙
1                              (结果)
? TG1 90↙
/                              (结果:不存在)
```

```
② TO TG2  : X
   IF OR ( : X=90) ( : X=270) [OP [/] STOP]
   MAKE "Z (SIN  : X) /COS  : X
   OP  : Z
   END
```

过程 TG2 也是正切函数,但只能在程序间调用,不能打印

输出(注意与 TG1 的区别)。

例 19. 余切函数。

```
TO CTG :X
IF OR (:X=0)(:X=180)[OP[/]STOP]
MAKE "Z (COS :X)/SIN :X
OP :Z
END
```

此函数过程只供程序间调用。

例 20. 指数函数。

```
TO ZS :N :Z :X :A
IF AND (:X=0)(:A=-1)[PR 1/ :M STOP]
IF AND (:X<1)(:A=1)[PR :M STOP]
IF AND (:X=0)(:A=0)[PR 1 STOP]
MAKE "M :N
ZS :N* :Z :Z :X- :A*1 :A
END
```

说明::N,:Z 均为底数,但:N 为变量,其终值即为指数函数的值;:A 为指数代码;当:X>0 时,即求"Z^X",则令:A=1,当:X=0 时,即求"Z^0",则令:A=0,当:X<0 时,即求"$1/Z^X$",则令:A=-1。在使用此过程时,需根据:X 值来确定:A 的值。

```
? ZS 5 5 3 1↙            (求 5³)
125                      (结果)
? ZS 5 5 -3 -1↙          (求 5⁻³)
8.N3
? ZS 5 5 0 0↙            (求 5⁰)
1
```

例 21. 求任意角的三角函数值及对应的弧度值。

```
TO SH :X :N :A
IF :X> :N [STOP]
MAKE "HD :X * 3.14/180
(PR [X=] :X[……HD=] :HD)
(PR [SIN X=] SIN :X[……COS X]COS :X)
(PR [TGX=] TG2 :X[……CTG X=]CTG :X)
PR [ ]
SH :X+ :A :N :A
END
```

说明：:X 为任意角的起始值，:N 为任意角范围内的最大值，:A 为打印角度间隔。如欲求从 0°～90°之间每隔 10°所对应的三角函数值，则 :X=0，:N=90，:A=10，请自己运行该程序。

例 22. 模拟硬币旋转后正面向上的次数，计算概率。

```
TO ZF :M :N :S
MAKE "Z RANDOM 2
IF :S<1 [MAKE "H :M+ :N (PR [M=] :M[ :N=]
   :N[ :GL=] :M/ :H) STOP]
IF :Z=1 [MAKE "M :M+1]
IF :Z=0 [MAKE "N :N+1]
ZF :M :N :S-1
END
```

说明：:M，:N 分别表示正反面出现的次数，其初值均为 0，:S 为试验次数。如：

```
? ZF 0 0 1000↙
M=504 :N=496 :GL=.504
```

可以看出，统计次数越多，概率越接近 0.5，并且要注意，在试验次数相同时，每次获得的概率不一定相同。

例 23. 用模拟圆形，求圆周率。

我们知道边数无限多的正多边形就是圆形，而正多边形的边长之和就是圆形的周长，圆的半径和正多边形的边长与边数

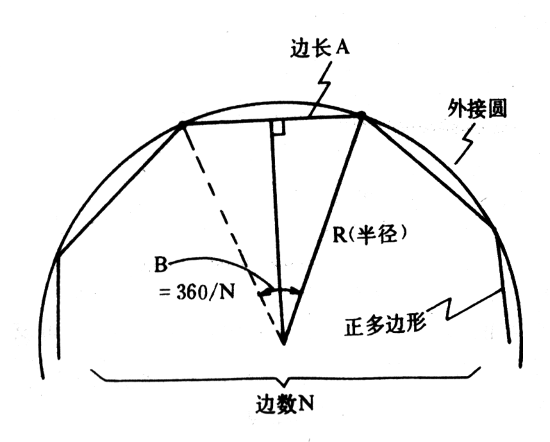


图 13.1　圆半径和正多边形的关系

的关系，由图 13.1 可知：

设边长 A，边数为 N，圆心角为 B，则有：B＝360/N，半径 R＝0.5A/SIN(B/2)，周长 S＝N・A 和直径 D＝2・R＝A/SIN(180/N)，则圆周率＝S/D＝N・SIN(180/N)

由以上关系，求圆周率的过程即可得：

```
TO YZ :N
MAKE "M :N * SIN 180/ :N
PR :M
END
? YZ 180↙
3.14143
```

例 24. 求连续整数的累加和。如求：

1＋2＋3＋……＋10＝？

```
TO QH :A :B :N
IF :B> :N [(PR :M [+] :M+1[+……+] :N[=] :A)STOP]
MAKE "A SUM :A :B
QH :A :B+1 :N
END


TO QHC :A :B :N
MAKE "M :A
QH :A :B :N
END
```

说明：:A，:B 的初值为最小的二位连续整数，:N 为连续整数中的最大值，过程 QHC 是为了保留:A 的初值而设，可以使打印结果便于浏览。如：

? QHC 12 13 1000↙　　（求 12+13+……+1000=?）

12+13+……+1000=500434　　（结果）

例 25. 求 N 的阶乘。求

N! =N×(N−1)×(N−2)×……×1。

```
TO JC :A :N
IF :N=1[(PR :N[! =]1)STOP]
IF :N=2[(PR :B[! =] :A)STOP]
MAKE "A :A*( :N-1)
JC :A :N-1
END


TO JCC :A :N
MAKE "B :A
JC :A :N
END
```

说明：:A，:N 的初值均为表达式的阶，但 :A 的值随 :N

递减的结果,即为：N(初值)的阶乘。如：

```
? JCC 10 10↙          (求 10×9×8×……×1=?)
10！=3628800              (结果)
```

例 26. 求一元二次方程的根。

一元二次方程表示为：$AX^2+BX+C=0$

求根公式为：$X1.2=(-B\pm\sqrt{B^2-4AC})/(2A)$

当 $B^2-4AC\begin{cases}>0,\text{方程有两个不等的实根。}\\=0,\text{方程有两个相等的实根。}\\<0,\text{方程的根为共轭虚根。}\end{cases}$

```
TO QG :A :B :C
MAKE "D :B* :B-4* :A* :C
IF :D>0 [MAKE "X1 (- :B+SQRT :D)/(2* :A)
         MAKE "X2 (- :B-SQRT :D)/(2* :A)
         (PR [X1=] :X1[…X2=] :X2)]
IF :D=0 [MAKE "X1 (-B)/(2* :A)
         MAKE "X2 :X1 (PR [X1=X2=] :X1)]
IF :D<0 [MAKE "X1 (- :B)/(2* :A) MAKE "X2 (SQRT - :
         D)/(2* :A)(PR [X=] :X1[+J] :X2)]
END
```

```
如:QG 3 9 -30↙
X1=2…X2=-5
? QG  1  -4  4↙
X1=X2=2
? QG  1  5  9↙
X=-2.5+J1.65831
```

例 27. 分解质因数。

```
TO QY :B :N :M
IF :M< :N [STOP]
```

```
MAKE "A REMAINDER  :M  :N
IF  :A>0 [MAKE "N  :N+1]
IF  :A=0 [MAKE "B  :B+1 (PR [Y] :B[=] :N)
          MAKE "M  :M/ :N]
QY  :B  :N  :M
END
```

说明：:N 的初值为最小除数 2，:M 为欲分解的任意正整数，:B 为质因数个数(初值为 0)。如：

```
? QY 0 2 136↙
Y1=2
Y2=2
Y3=2
Y4=17
```

例 28. 求素数。

```
TO QS  :B  :N  :M  :H
IF  :M> :H [STOP]
IF  :N= :M [MAKE "B  :B+1 (PR [S]  :B[=] :M)]
MAKE "A REMAINDER  :M  :N
IF  :A>0[MAKE "N  :N+1]
IF  :A=0 [MAKE "M  :M+1 MAKE "N 2]
QS  :B  :N  :M  :H
END
```

说明：:M，:H 为所求数值范围内的最小值和最大值，:N 仍为最小除数 2。如：

```
? QS 0 2 3 100↙
S1=3
S2=5
S3=7
```

S4＝11

⋮…

S24＝97

例 29. 打印闰年年号。

我们知道平年有 365 天，闰年有 366 天，其规律是：

①年号能被 4 除尽，不能被 100 除尽的是闰年；

②年号能被 400 除尽的是闰年。

根据上述规律，编制过程如下：

```
TO YR :X :Z
IF :X> :Z [STOP]
MAKE "M REMAINDER :X 4
MAKE "N REMAINDER :X 100
MAKE "K REMAINDER :X 400
IF OR (AND (:M=0)(:N>0))(:K=0)[PR :X]
YR :X+1 :Z
END
```

说明：:X :Z 分别起止年号。如：

```
? YR 1989 2000
1992
1996
2000
```

13.5 字和字处理

1. 什么是字？

由字母或数字等符号组成的一串字符称为“字”。字的标志是在串字符前加双引号“"”。如"APPLE，"LOGO，"X5，"135 等。需要注意的是：LOGO 中的字不同于 BASIC 中的字符串，因为 LO-

GO 的字中不允许有空格，括号和各运算符。如"B 3，"A＋B，"A *
2，"F(X)等均不是字。

对于字，除可用打印命令“**PR**”输出外，还有专门的字处理命令。

2. 字的处理

格式 1：**FIRST**〈字〉

作用：取出字的第一个字符。

例 30.

? PR FIRST "APPLE↙

A

格式 2：**LAST**〈字〉

作用：取出字的最后那个字符。

例 31.

? PR LAST "APPLE↙

E

格式 3：**BUTFIRST/BF**〈字〉

作用：取出字中除第一个字符以外的其它字符。

例 32.

? PR BF "APPLE↙

PPLE

格式 4：**BUTLAST/BL**〈字〉

作用：取出字中除最后那个字符以外的其它字符。

例 33.

? PR BL "APPLE

APPL

如果一个字中只含有一个字符，那末取出其第一个(最后一个)字符后为空。如：

? PR BF "X↙,屏幕上显示一个空行,表示打印输出的是一个“空字”(即不含任何字符的字)。

格式 5:**WORD** 〈字 1〉〈字 2〉…〈字 n〉

作用:将给定的字连接,从而生成一个新的字。

例 34.

? PR WORD "ABC   "DEF↙

ABCDEF

注意:

三个或三个以上的字连接时,需用圆括号将命令和所有的字括在一起。

13.6 表和表处理

1. 什么是表?

把若干个字组合在一起,并用方括号[ ]括起来,就组成一个“表”。表中的字不再带双引号,每个字(又称表元素)彼此间要用空格或运算符号隔开。如:[APPLE LOGO],[ZHJ SYDQ],及[A+B=]等均是正确的表。这里的方括号可以看成是表的标志,它不是表中内容,即当打印一个表时,方括号不会被打印出来。

2. 表的处理

对于表处理,字处理中的前四条命令同样适用于表处理,只是将字符换成表元素而已。

例 35.

? PR FIRST [APPLE LOGO]↙

APPLE

? PR LAST [APPLE LOGO]↙

LOGO

? PR BF [ZHJ SYDQ LOGO]↙

SYDQ LOGO

? PR BL [ZHJ SYDQ LOGO]↙

ZHJ SYDQ

表处理中表的连接命令与字处理不同。

格式:**SENTENCE/SE**〈字/表/数〉

作用:将若干字、数或表进行连接,构成新表。

例 36.

? PR (SE [ZHJ] [SYDQI]"1990)↙

ZHJ SYDQ 1990

对于表处理,还有另外两个命令:

格式:**FPUT**〈字/表〉〈表〉

作用:将字或表作为一个表元素放在后表中元素的最前面,组成一个新表。

例 37.

? PR FPUT "APPLE [—LOGO]↙

APPLE—LOGO

? PR FPUT [ZHJ] [SYDQ]↙

[ZHJ]SYDQ

格式:**LPUT**〈字/表〉〈表〉

作用:将字或表作为一个表元素放在后表中所有元素之后,组成一个新表。

例 38.

? PR LPUT "1990 [ZHJ SYDQ]

ZHJ SYDQ 1990

? PR LPUT [ZHJ] [BASIC LOGO]

BASIC LOGO [ZHJ]

13.7 字表处理应用举例

例 39. 将一个字符串中字母从右到左依次去掉。

```
TO ZL  :N
IF :N="  [PR "     STOP]
PR :N
ZL BL :N
END
? ZL "ABCDEFG
ABCDEFG
ABCDEF
ABCDE
ABCD
ABC
AB
A
```

例 40. 打印表元素的字头(缩写)。

```
TO QW :D
IF :D=[ ][STOP]
MAKE "A WORD :A FIRST(FIRST :D)
QW BF :D
END
TO PL :D
MAKE "A BF "X
QW :D
PR :A
END
```

说明:过程 PL 中调用了过程 QW,:D 为一个表。

? PL [BEGINNER′ S ALLPURPOSE SYMBOLIC INSTRUCTION CODE]↙

BASIC

例 41. 将蜡烛下的三个圆盘由 A 位 13.2(a)移到 B 位 13.2(b)，如图 13.2 所示。移动时注意：

①每次只能移动一个圆盘，而且只能从上面取盘。

②移动过程中要求在任何时候，小盘总是放在较大的盘上。

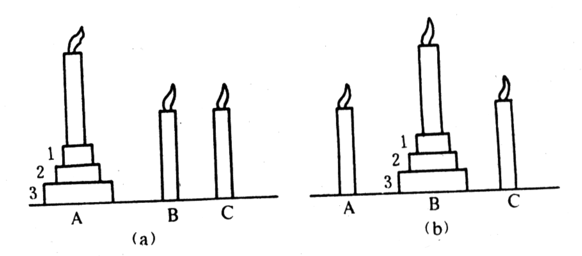


图 13.2　移动圆盘示意图

下面是解决这个移动问题的程序：

```
TO YP :N :X :Y :Z
IF :N=0 [STOP]
YP :N-1 :X :Z :Y
(PR [P] :N :X [→] :Y)
YP :N-1 :Z :Y :X
END
```

说明：:N 为圆盘总数，:X，:Y，:Z 表示位置，运行时用对应的字"A，"B，"C 代替，符号"→"表示移动方向，P1～P3 表示几号盘。

现设：N=3，键入：

? YP 3 "A "B "C↙

P1 A→B

P2 A→C

P1 B→C

P3 A→B

P1 C→A

P2 C→B

P1 A→B

至此已按要求移动完毕。可以证明，移动次数 M 与盘数 N 的关系为：$M = 2^N - 1$。

这是世界上有名的梵塔(Hanoi)问题。关于这个问题还有一个神奇的传说：相传在古代印度的布拉玛婆罗门圣庙里的僧侣正在进行一种称为“梵塔”的游戏。这个游戏是：

一块黄铜平板上有三根钻石做的细柱 i、j、k，i 柱上套着从上到下，由小到大，编号为 1，2……64 的黄金圆盘。游戏要求把 i 柱上的圆盘全部移到 k 柱上，仍按同样顺序叠成塔形，且移动时必须遵守：每次只许移动最上面的一个圆盘，且不许把大盘压在小盘上。主神梵天宣称：如果人们不对庙里的神灵顶礼膜拜，当他们做完这个游戏时，世界将在一声霹雳声中毁灭，而那些虔诚的信徒都可以升天。

可以证明，要移动完这 64 个圆盘，至少需要移动圆盘 $2^{64}-1$ 次。假设人们每秒钟移动一次，要作完这个游戏，就需要大约 5800 亿年。所以主神的宣称，完全是骗人的。

习题十三

1. 试自定义如下函数：

①$Y = X^2$；

②$Y=1/X$；

③$Y=\sqrt{A^2+B^2}$

2. 设计一个求解一元一次方程 $AX+B=0$ 的根的过程。

3. 设计一个求 A/B=C……D 的过程。其中 A,B 为整数，且 A>B,C 为整数商，D 为余数。如：23/4=5……3。

4. 设计一个求任意正整数 A,B,C 的最小公倍数的过程。(为简化过程设计，A,B,C 可按大小顺序排列)。

5. 设计一个求“1/2+1/3+1/4+……+1/10=？”过程。

6. 按例 41 的要求，亲自做一下这个梵塔游戏，取 N=4 试一试看，至少需要移动多少次？

第十四章　函数图形的程序设计

迄今为止，我们见到的用 LOGO 语言来绘制的图形大多是一些几何图形。本章将介绍另一类图形——函数图形的绘制方法和编程技巧。有关这些内容的资料目前尚不多见。

14.1　哪些函数能够用来绘图

鉴于屏幕为平面直角坐标，故海龟绘图对应的函数是二元函数。凡是二元函数，次数不限，原则上都可以由海龟来绘出函数图形（指数函数图形除外，因 LOGO 没有指数函数的运算）。而且当改变函数中某些参数的值时，还可以通过图形的变化，来分析函数的特征。

能够用 LOGO 来绘出图形的函数，归纳起来有以下几种表现形式：

1. $Y=f(x)$

如：$Y=A\ SIN\ X$；

$Y=8A^3/(X^2+4A^2)$等。

这种函数的自变量可以直接赋值，变量的值由函数关系确定。

2. $\begin{cases} X=f_1(T) \\ Y=f_2(T) \end{cases}$

这是用参数方程表示的函数，如：

$$\begin{cases} X = A \cdot \text{COS } T \\ Y = B \cdot \text{SIN } T \end{cases}$$

$$\begin{cases} X = A(\text{COST} + T \cdot \text{SIN } T) \\ Y = A(\text{SIN } T - T \cdot \text{COS } T) \end{cases}$$等。

用这种方式表达的函数，只需对 X 和 Y 分别按对应的关系赋值，自变量取参变量 T。

3. $\rho = f(\theta)$

这是用极坐标方式表示的函数，如：

$\rho^2 = A^2 \cdot \text{COS}2\theta$

$\rho = A \cdot \text{SIN}3\theta$ 等。

这种方式的函数，在绘图时，应按以下关系转换成参数方程，然后在分别给予赋值。

转换的公式为：$\begin{cases} X = \rho \cdot \text{COS}\varphi \\ Y = \rho \cdot \text{SIN}\varphi \end{cases}$

其中 φ 为 ρ 与横轴的夹角。

如：$\rho = A \cdot \text{SIN}3\theta$ 应该转换成：

$$\begin{cases} X = \rho \cdot \text{COS}\varphi = A \cdot \text{SIN}3\theta \cdot \text{COS}\varphi \\ Y = \rho \cdot \text{SIN}\varphi = A \cdot \text{SIN}3\theta \cdot \text{SIN}\varphi \end{cases}$$

自变量取 θ，角 φ 一般为常数。

14.2 函数绘图程序的结构

TO〈过程名〉〈参数 1〉〈参数 2〉……〈参数 N〉

〈条件语句〉
〈赋值语句〉
〈绘图语句〉
〈递归语句〉} 过程体

END

过程体中至少包括这四个方面的语句。下面就每一类语句的特点及如何设置，分别说明。

14.2.1 条件语句

条件语句的作用有两个：一是使递归过程停止；二是为下次绘图作准备，使海龟复位。当然，要作到这二点并不困难，它可以由命令序列完成，关键在于应该满足什么样的条件，才会停止并复位呢？

以绘制正弦函数曲线为例，当自变量 X 变化一个周期时，就可以画出一个正弦波形。显然，绘制图形的个数或者自变量(包括参变量)的变化范围，都可以作为继续或停止的条件，如绘制正弦曲线的条件语句可以写成：

IF :X>360[STOP]

这里正弦函数只显示出一个波形，即停止。若再老虑到复位的情况，则条件语句可为：

IF :X>360[PU SETPOS [X0 Y0]PD STOP]

其中 PU SETPOS [X0 Y0]PD 即为复位命令，其作用是在过程停止运行之前，将海龟调回到函数图形绘制的起始位置，其坐标为[X0 Y0]。

一般说来，条件就取自变量值允许的变动范围。

14.2.2 赋值语句

1. 如何确定自变量的初值

自变量的赋值，是指其初值的确定，因为整个绘图过程是由自变量的初值开始，经过递增语句来完成的，自变量的最大值由条件语句限定，故关键在于确定初始值。

表面上看，令自变量初值为0就是了，但这里有二个问题：一个是自变量等于0时在屏幕上的位置；二是自变量单位与坐标单位之间存在着差异。

①自变量等于0的位置

为了在整个屏幕上显示完整的对称图形，我们希望图形起点坐标不应该在屏幕坐标原点上，而应该尽量靠近屏幕左侧，这样当自变量增加时，使整个图形从左到右显示在屏幕上。

我们已经知道，屏幕横轴的坐标范围是从－140到＋139，考虑到坐标系占去的位置，可取横坐标－130处左右作为图形起始点坐标。对于与原点对称的图形，起始点坐标可取(－130，0)。为使第Ⅰ象限的图形在整个屏幕上显示出来，起始点坐标可取为(－130，－75)，相当于将坐标原点移到屏幕的左下角。

②坐标单位与函数单位的换算

设自变量X的变动范围为0～360，则其对应的屏幕坐标范围应从－130到(－130＋360)，即X应在－130和＋230之间变化。但屏幕横坐标的正向最大值是＋139，显然X的变动范围超出了屏幕范围，因此需要进行单位换算。

我们可以首先把允许X坐标变动的范围确定下来，比如说让X只能在－130到＋70之间变化，则变动范围为200个海龟步。对于正弦曲线来讲，希望在这200个海龟步中使X由0增加到360，显然每步对应的X值为：360/200＝1.8，则当X按360/1.8的关系变动时，其值为360时，恰好走了200个海龟步。这就是单位换算。

由此可得，正弦函数自变量的赋值语句为：

MAKE "X :X/1.8－130，:X的初值为0，由键盘在运行时输入。不难看出，这样赋值后，当X＝0时，其位置在－130处，当X＝360时，其位置恰好在＋70处。

2. 变量的赋值

变量的赋值可以直接按对应的包括自变量在内的函数关系赋值。

如：MAKE "Y ：A * SIN ：X，其中：A 为正弦函数的幅值。其值大小以不超过屏幕上纵坐标的范围为限。

在实际编写程序时，为增加程序的变化，可适当增加一些参数，如：

MAKE "Y 80 * ：AM * SIN ：X * ：A + ：AW

其中"80"为正弦量幅值，：AM 为幅值系数，其值取大于 0 小于或等于 1，：A 为频率系数，当：A 的值大于 1 时，屏幕上显示的波形数将大于 1，：AW 为相位。

经过这样处理后的赋值语句，可以使我们获得任意幅值、频率和相位的正弦曲线，从而大大扩展了程序的绘图功能。

14.2.3 绘图语句

在给自变量和变量赋值以后，即已有了：X 和：Y，则绘图语句可直接由纵横坐标绘图命令组成，即 SETPOS SE ：X ：Y。

14.2.4 递归语句

在过程体中，调用过程自身的语句称递归语句，在递归语句中对应自变量的那个形式参数要加上一个适当的增量。增量的大小，应与函数关系相对应，其值大，绘图速度快，但图形粗糙，其值小，所绘图形较准确，但绘图时间就长。鉴于此，通常将增量取为自变量最大值的百分之一或二较为适宜。对于正弦函数，自变量：X 的增量可取 2～5，即：X+2 或：X+5。

这里，以正弦函数为例，介绍了编制函数绘图程序 的步骤和要求。在此，我们给出一个绘制正弦函数的完整程序。

例 1. 画正弦函数图形。

```
TO ZX :X :AM :A :AW
IF :X>360 [PU SETPOS [-130 0] PD STOP]
MAKE "X :X/1.8-130
MAKE "Y 80* :AM*SIN :X* :A+ :AW
SETPOS SE :X :Y
ZX :X+2 :AM :A :AW
END
```

下面运行一下此过程，请键入：

? PU SETPOS [-130 0] PD↙

? ZX 0 1 1 0↙　　(曲线①)

注意观察海龟移动绘图的全过程及复位的情况。然后再键入：

? ZX 0 1 1 -120↙　　(曲线②)

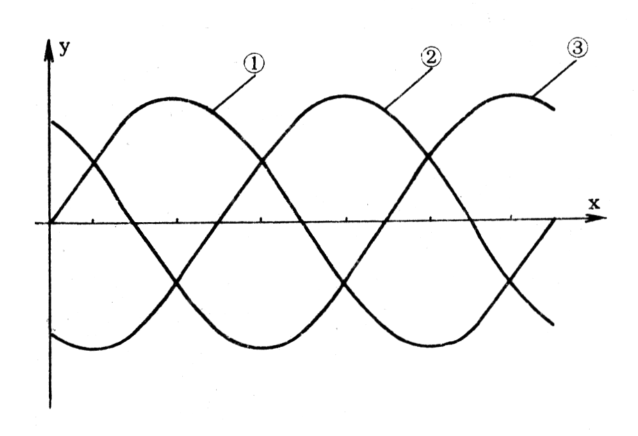


图 14.1　三相对称正弦曲线

? ZX 0 1 1 120↙　　(曲线③)

这就在屏幕上得到三条幅值相等，频率相同，相位互差120°的正弦曲线，这种曲线，称三相对称正弦曲线。若再为曲线画出直角坐标系，则如图 14.1 所示。

读者还可给出其它参数值进行试验，参数值最好按以下范围选取：0≤：AM≤1.2，1≤：A≤10，0≤：AW≤360。

若将正弦过程稍加修改，可以获得意想不到的结果。修改的过程见习题十四，读者不妨按题目要求运行，并加以比较。

14.3 函数图形程序选编

例 2. 条件图表。

```
TO TB :X :A
IF OR (:X>240)(:A>30)[SETX -120 STOP]
MAKE "Y RANDOM 160
PU SETPOS SE :X-120 -60 PD
REPEAT 3[FD :Y+10 BK :Y+10 RT 90 FD 1 LT 90]
TB :X+240/(:A-1):A
END
```

说明：此过程无实际物理意义，仅为训练编程而设计。如：

？TB 0 9↙，显示如图 14.2

例 3. 模拟分子的热运动(布朗运动)。

```
TO FV :N
IF :N<1 [BK 30 RT 90 REPEAT 36[FD .3 RT 10] PU HOME PD
        STOP]
MAKE "X RANDOM 240
MAKE "Y RANDOM 200
SETPOS SE :X-120 :Y-100
WAIT 30
```

```
FV :N-1
END
```

说明：:N 为运动的轨迹数。如：

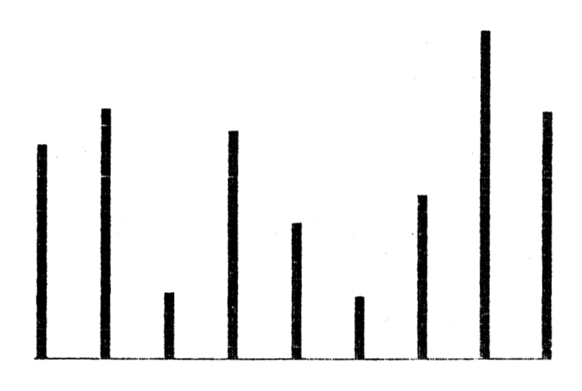


图 14.2　图表

? FV 20↙，显示如图 14.3。

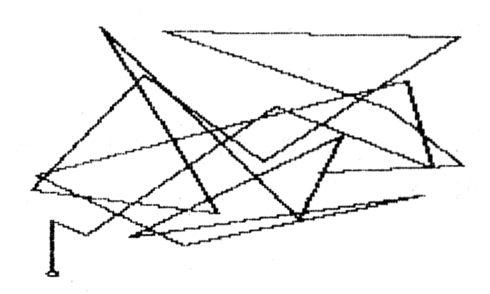


图 14.3　布朗运动图示

例 4. 繁星闪烁。

```
TO MX :N
IF :N<1 [PU HOME PD STOP]
MAKE "X RANDOM 240
```

```
MAKE "Y RANDOM 200
DOT SE :X-120 :Y-100
WAIT 20
SETPC 0 DOT SE :X-120 :Y-100
WAIT 20
SETPC 1 DOT SE :X-120 :Y-100
MX :N-1
END
```

说明：

①此过程基本上同例 3，仅用坐标打点命令替换了坐标绘图命令。

②通过“WAIT”的延时作用和彩色绘图命令的擦抹功能，可以使程序在运行中出现星光闪烁的效果。如：

? MX 50↙，显示如图 14.4

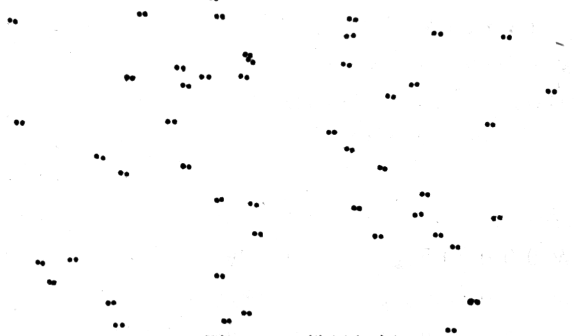


图 14.4　繁星闪烁

例 5. 圆和椭圆。

```
TO TY :X :Y :Q :A :B
IF :Q>360 [PU HOME PD STOP]
MAKE "M :X+ :A * COS :Q
MAKE "N :Y+ :B * SIN :Q
```

```
SETPOS SE :M :N
TY :X :Y :Q+5 :A :B
END
```

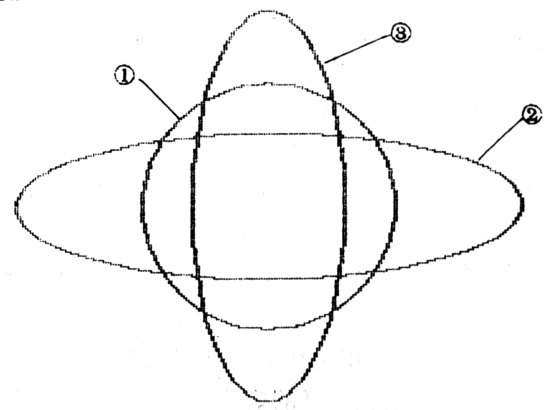


图 14.5 圆和椭圆

说明：:A 为椭圆横轴，:B 为椭圆纵轴，当 :A= :B 时，即为一个圆。(:X，:Y)为圆心坐标。

```
TO YW :X :Y :Q :A :B
PU SETPOS SE :X+:A :Y PD
TY :X :Y :Q :A :B
END
```

请键入：

? YW 0 0 0 50 50↙ (曲线①)

? YW 0 0 0 100 30↙ (曲线②)

? YW 0 0 0 30 80↙ (曲线③)

结果显示如图 14.5。

例 6. 上面所绘之椭圆的长短轴都是与坐标轴平行的。若绘制轴倾斜的椭圆，需经过坐标系变换，请看下面过程：

```
TO KJ :X :Y :Q :A :B :T
IF :Q>360 [PU HOME PD STOP]
```

```
MAKE "M :X+ :A*(COS :Q)*(COS :T)- :B*
       (SIN :Q)*SIN :T
MAKE "N :Y+ :A*(COS :Q)*(SIN :T)+ :B*
       (SIN :Q)*COS :T
SETPOS SE :M :N
KJ :X :Y :Q+5 :A :B :T
END
```

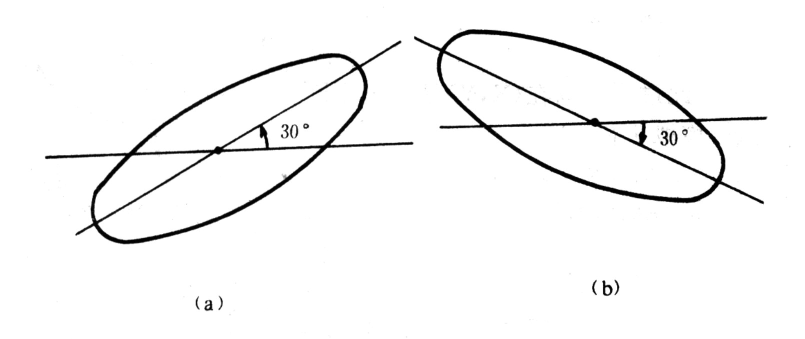


图 14.6　倾斜椭圆

说明：:T 为椭圆横轴倾斜的角度：

:T>0↙，:T<0↙

```
TO KH :X :Y :Q :A :B :T
MAKE "H :X+ :A*COS :T
MAKE "Z :Y+ :A*SIN :T
PU SETPOS SE :H :Z PD
KJ :X :Y :Q :A :B :T
END
? KH -70 50 0 50 20 30↙
? KH 70 50 0 50 20 -30↙
```

运行结果分别见图 14.6(a)(b)。

例 7. 若再加上下面二个过程，可以绘出如图 14.7 所示的“科技之花”的图案。

```
TO HX :X :Y :A
PU SETPOS SE :X :Y PD
RT 90 BK 1
REPEAT 4[FD 2 LT 90] FD 1 LT 90
PU BK :A+60 PD
END
```

此过程可以在(X，Y)处画一小圆点，模拟花芯。

```
TO YH :N :R :A
REPEAT :N[FD :R * 3.14/18 RT 5 * :A]
END
```

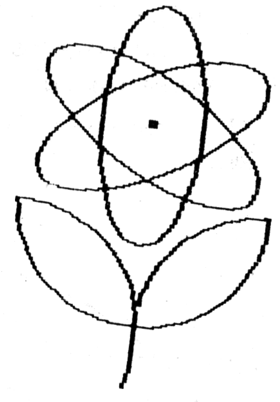


图 14.7 科技之花

此过程为画圆弧过程，见 14.11.4。

```
? KH 0 30 0 50 20 30↙
? KH 0 30 0 50 20 -30↙
? KH 0 30 0 50 20 90↙
? HX 0 30 50 LT 10 YH 3 50 1↙
? YH 18 25 1 RT 90↙
? YH 36 25 1 RT 90↙
? YH 16 25 1↙
```

上述过程运行完毕，即得“科技之花”。

例 8. 弹道曲线。

弹道曲线的方程为：

$$\begin{cases} x = v_0 t cos\alpha \\ y = v_0 t sin\alpha - \frac{1}{2} g t^2 \end{cases}$$

请看下面过程：

```
TO DD :T :V :Q
MAKE "A :V * :T * COS :Q
MAKE "B :V * :T * (SIN :Q) - 4.9 * :T * :T
SETPOS SE :A - 130 :B - 75
IF :B < 0 [PU SETPOS [-130 -75] PD STOP]
DD :T + .1 :V :Q
END
```

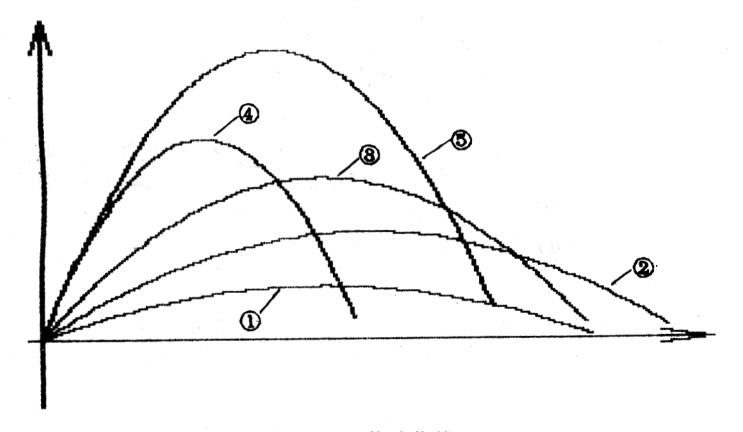


图 14.8 弹道曲线

说明：:T 为时间，初值为 0；:V 为初速，:Q 为仰角（海龟方向与横轴正方向夹角）。

再为其配上直角坐标系（略），即可以进行“射击”表演了。

? DD 0 50 30↙ （曲线①）

? DD 0 50 45↙ （曲线②）

? DD 0 50 60↙ （曲线③）

? DD 0 50 75↙ （曲线④）

? DD 0 60 75↙　　　　　　　　　(曲线⑤)

显示如图 14.8。由图中可以看出：

①初速相同时，仰角为 45°时，射程最远。

②仰角相同时，初速越大，射程越远。

例 9. 玫瑰线。

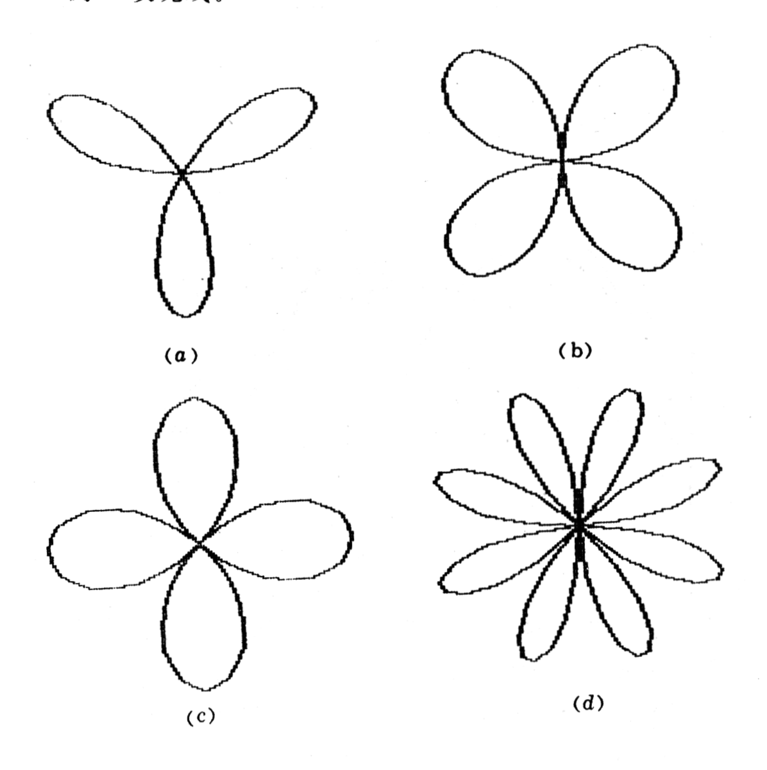


图 14.9　玫瑰线

函数表达式为：$\rho = A \cdot SIN(N\theta + \beta)$

TO MG :Q :A :W :N :H :Z

```
IF :Q>360 [PU HOME PD STOP]
MAKE "X :H+ :A * (SIN :Q * :N+ :W) * COS :Q+ :W
MAKE "Y :Z+ :A * (SIN :Q * :N+ :W) * SIN :Q+ :W
SETPOS SE :X :Y
MG :Q+5 :A :W :N :H :Z
END
```

说明：:H，:Z 为图形中心位置坐标，:N 为玫瑰线叶片数代码：:N=3 表示三叶玫瑰线，:N=2 表示四叶玫瑰线，:N=4 表示八叶玫瑰线。

```
TO MGX :Q :A :W :N :H :Z
MAKE "H1 :H+ :A * (SIN :Q * :N+ :W) * COS :W
MAKE "Z1 :Z+ :A * (SIN :Q * :N+ :W) * SIN :W
PU SETPOS SE :H1 :Z1 PD
MG :Q :A :W :N :H :Z
END
? MGX 0 60 0 3 -75 30↙    (图(a))
? MGX 0 60 0 2 75 30↙     (图(b))
? MGX 0 60 90 2 -75 10↙   (图(c))
? MGX 0 60 0 4 75 10↙     (图(d))
```

显示图形如图 14.9(a)～(d)。

例 10. 调幅波(模拟无线电波)。

```
TO TF :X :A :Z
IF :X>540 [PU SETPOS [-130 0]PD STOP]
MAKE "H :X/2.7-130
MAKE "M :Z * (60+20 * SIN :X)
MAKE "N SIN :X * :A+90
MAKE "Y :M * :N
SETPOS SE :H :Y
TF :X+4- :A/5 :A :Z
```

END

说明：:A 为载波频率大于音频波频率的倍数，:Z 为绘制音频波代码：:Z 取 1 或者－1。

再加上直角坐标系(略),运行此过程：

? TF 0 6 1↙　　(曲线①)

? TF 0 0 1↙　　(曲线②)

? TF 0 0 -1↙　　(曲线③)

显示如图 14.10 所示。

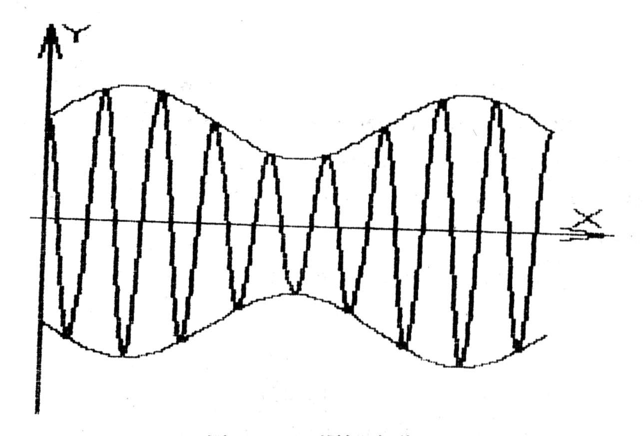


图 14.10　调幅波形

例 11. 箕舌线。

函数表达式为：$Y=8A^3/(X^2+4A^2)$

```
TO JS :X :A
IF :X>120 [PU HOME PD RT 90 BK 2 REPEAT 36 [FD :A*3.14/
        36 LT 5 PU FD :A*3.14/36 LT 5 PD] LT 90 STOP]
MAKE "X :X
MAKE "Y 8*:A*:A*:A/(:X*:X+4*:A*:A)
SETPOS SE :X :Y
JS :X+4 :A
```

```
END
TO JSZ :X :A
SETX 120 SETX -120
PU SETY 8*:A*:A*:A/(120*120+4*:A*:A)
        PD
JS :X :A
END
? JSZ -120 40↙          (曲线①)
? JSZ -120 25↙          (曲线②)
```

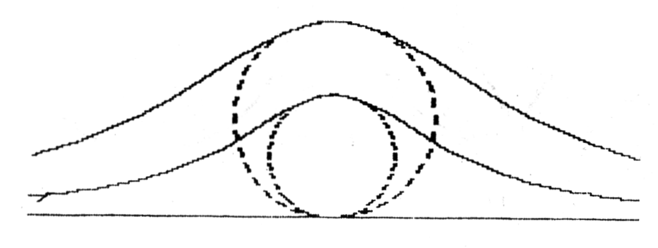


图 14.11　箕舌线

显示如图 14.11。

例 12. 星形线。

函数表达式为：$\begin{cases} X = A \cdot COS^3t \\ Y = A \cdot SIN^3t \end{cases}$

```
TO XX :T :A
IF :T>360 [BK 2 REPEAT 36[FD :A*3.14/36 LT 5 PU FD :A*
      3.14/36 LT 5 PD]PU HOME PD STOP]
MAKE "X :A*(COS :T)*(COS :T)*(COS :T)
MAKE "Y :A*(SIN :T)*(SIN :T)*(SIN :T)
SETPOS SE :X :Y
XX :T+4 :A
END
TO XXZ :T :A
```

```
PU RT 90 FD :A LT 90 PD
XX :T :A
END
? XXZ 0 60 ↙            (曲线①)
? XXZ 0 35 ↙            (曲线②)
```

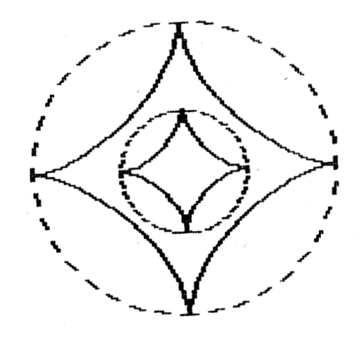


图 14.12 星形线　　　　图 14.13 心脏线

显示结果如图 14.12。

例 13. 心脏线。

函数表达式为：$\rho = A \cdot (1 - COS\alpha)$

```
TO XZ :Q :A :W
IF :Q>360 [REPEAT 2[REPEAT 54[FD :A*3.14/72 LT 5 PU FD
        :A*3.14/72 LT 5 PD] LT 180 BK 2 ]STOP]
MAKE "X :A*(1-COS :Q)*COS :Q+ :W
MAKE "Y :A*(1-COS :Q)*SIN :Q+ :W
SETPOS SE :X :Y
XZ :Q+4 :A :W
END
? XZ 0 45 0↙
```

显示结果如图 14.13。

实际上，我们还可以利用 LOGO 语言画出许多有实用价值

的函数曲线，读者有兴趣的话，不妨自己试一试。

习题十四

1. 运行下述程序，改变参数值，观察图形。

```
TO ZXH :X :AM :A :AW :BM :B :BW
IF :X>360 [PU SETPOS [-130 0]PD STOP]
MAKE "X :X/1.8-130
MAKE "M 80* :AM*SIN :X* :A+ :AW
MAKE "N 80* :BM *SIN :X* :B+ :BW
MAKE "Y :M+ :N
SETPOS SE :X :Y
ZXH :X+2 :AM :A :AW :BM :B :BW
END
```

其中各参数使用范围见例 1。

2. 请使用椭圆过程和随机函数在屏幕上绘出若干个位置不同的汽球，看谁画得好。

3. 圆的渐开线方程为：

$$\begin{cases} X=A(COST+T\cdot SINT) \\ Y=A(SINT-T\cdot COST) \end{cases}$$

试编写一过程，绘出渐开线。

提示：公式中的"T"换算成弧度。

4. 阿基米德螺线方程为：$\rho=A\cdot\theta$

试编写一过程，绘出螺线。

提示：θ 为转角，取弧度单位。

附录一　BASIC 语言一览表

这里列出了 BASIC 中要用到的命令、函数以及在 BASIC 状态下可用的 DOS 命令。

1.1　BASIC 命令一览表

CALL〈**算术表达式**〉

调用机器语言子程序。算术表达式的值必须在－65535 和 65535 之间。

COLOR＝〈n〉

设置低分辨率图形显示的颜色。n 是颜色的代号，其对应关系如下：

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
颜色	黑	红	深兰	紫	深绿	灰 1	中兰	浅兰	棕	橙	灰 2	粉红	浅绿	黄	兰绿	白

DATA〈**数据**〉[,〈**数据**〉,…]

为 READ 语句提供数据。

DEF FN〈**变量名**〉(〈**实型算术变量**〉)＝〈**算术表达式**〉

说明自定义函数。该函数一经定义，就可以象标准函数 SIN(X)等一样在算术表达式中调用。调用形式为 FN〈变量名〉(〈实在参数〉)。

DIM〈**变量名**〉(〈**下标表**〉)[,〈**变量名**〉(〈**下标表**〉)]

定义一个或多个数组。下标表由一个或多个下标构成。下

标的值称为数组的界。下界为 0，上界最大为 32767。下标的个数称为数组的维数，数组的最大维数是 88。常用的是一、二、三维数组。数组也有类型之分。$ 代表字符型，% 表示整型。

DRAW n[AT X,Y]

在屏幕上显示第 n 个向量表中的图形。

FLASH

置显示器为闪烁方式。中文 BASIC 无效。

FOR 〈**实型算术变量**〉=〈**算术表达式** 1〉 TO 〈**算术表达式** 2〉[STEP 〈**算术表达式** 3〉] 〈**循环体**〉 NEXT 〈**实型变量名**〉

循环执行 FOR 和 NEXT 之间的语句(即循环体)，直到循环控制变量的值超过终限值，则继续 NEXT 后继命令的执行。循环体中又可以嵌套循环，但最多只能嵌套 10 层。

FRE(X)

释放数据区中的无用单元。X 仅是一个"虚"元。可取为 0。

GET 〈**变量名**〉

从当前输入设备(键盘)上接收一个字符并送到变量中。执行 GET 时，屏幕上没有任何提示信息，输入的字符也不显示在屏幕上。由于仅接收一个字符，因而敲入字符之后可以不敲回车键。

GOSUB 〈**行号**〉

无条件转到给定行号的子程序去执行，并将 GOSUB 的下一个语句地址推入堆栈，以便子程序执行结束到 RETURN 时返回到该地址继续执行。子程序嵌套调用最多 25 层。

GOTO 〈**行号**〉

无条件转到给定的行号去执行。

GR

置低分辨率作图方式。这个语句是将屏幕上部置成 40×40 可显示的小方块，作为低分辨率图象显示区，将屏幕下部四行作为文本显示区。如果要将这种方式转换到纯低分辨图形显示，那么可用 POKE 或 PEEK 语句去访问－16300。

HCOLOR＝〈n〉

设置高分辨率图形的颜色。在高分辨图形方式下有八种颜色，其对应关系如下表：

n 值	0	1	2	3	4	5	6	7
颜色	黑 1	绿	兰	白 1	黑 2	红	黄	白 2

HGR

设置屏幕高分辨率图形与文本显示混合方式。显示第一页。

HGR2

作用同 HGR，仅使用第二页。访问地址－16302 可转到纯高分辨图象显示方式，访问－16301 又可转回来。

HIMEM:X

设置 BASIC 程序工作区域的上限地址，目的是防止 BASIC 程序的数据破坏高分辨图形显示区域机器语言子程序。

HLIN X1,X2 AT Y

在低分辨率作图方式下，以(X1,Y)为起点，(X2,Y)为终点在屏幕上画一条水平线(横线)。

HPLOT X,Y

在高分辨图形方式下以(X,Y)为座标画一个点。

HPLOT TO X,Y

从上次所画点到(X,Y)座标之间画一条线。

HPLOT X1,Y1 TO X2,Y2 [{TO X_i,Y_i}…]

从(X1,Y1)到(X2,Y2)之间画一条线,再从(X2,Y2)到(X3,Y3)之间画一条线,依次画下去,使所有的(X_i,Y_i)连成一条线。其中 0≤X≤279,0≤Y≤159。如果是在满屏方式下,则 0≤Y≤191。

HPLOT TO X2,Y2 [{TO X_i,Y_i}…]

从上次所画点到(X2,Y2)之间画一条线,再依次连到(X_i,Y_i),这样一直画下去。

HTAB (〈算术表达式〉)

将光标移到指定的列上。在西文 BASIC 下,算术表达式的值必须在 1～255 之间。从光标所在行的 1～40 列为当前行,41～80 列为下一行;在中文 BASIC 下,算术表达式的值也必须在 1～255 之间。1～34 为当前行,35～68 为下一行。

IF 〈**条件**〉 GOTO 〈**行号**〉

如果条件为真,则转到指定行号去执行。否则顺序执行 IF 的下一个语句。

IF 〈**条件**〉 THEN 〈**行号**〉

如果条件为真,则转到行号所指语句去执行,否则顺序执行下一语句。

IF 〈**条件**〉 THEN 〈**语句**〉[:〈**语句**〉…]

如果条件为真,则执行 THEN 之后给出的语句,否则执行 IF 的后继语句。

INPUT [〈**提示**〉;]**变量名** [{,〈**变量名**〉}…]

允许从键盘上接收数据并送到变量中。

INVERSE

将屏幕置为反相显示,即白底黑字。

LET 〈**变量名**〉=〈**表达式**〉

将表达式的值赋给左部变量。LET 可以省略。

LOMEM:X

限制 BASIC 程序工作区域的下限地址。参见 HIMEM。

MUSIC X,Y

定义一个音节和音长,并使扬声器发声。其中 X 表示发声的频率,Y 表示发声的时间。

X 对应的音阶如下表:

X 值	255	228	205	192	171	152	140	128	114	102	95	84	75	68	62
音阶	$\underset{\cdot}{5}$	$\underset{\cdot}{6}$	$\underset{\cdot}{7}$	1	2	3	4	5	6	7	$\dot{1}$	$\dot{2}$	$\dot{3}$	$\dot{4}$	$\dot{5}$

Y 对应的节拍如下:

Y 值	30	70	110	160	255
音长	1/4 拍	1/2 拍	1 拍	拍 2	4 拍

NORMAL

把显示器置为标准(正常)方式,即黑底白字。

NOTRACE 取消 TRACE 的跟踪作用。

ON 〈n〉 GOSUB 〈行号 1〉,〈行号 2〉[,〈行号 3〉…]

根据 n 的值,转到第 n 个行号(即行号 n)的子程序去执行。

ON〈n〉GOTO〈行号 1〉,〈行号 2〉[,〈行号 3〉…]

根据 n 的值,转到行号 n 的语句去执行。

ONERR GOTO 〈**行号**〉

如果程序发生错误则无条件转到指定行号去执行。

PLAY

将盒式磁带机中游戏软件装 入内存自动运行。操作方法是:

先将磁带退到开始位置,在键盘上敲入 PLAY,按下磁带机

上的“PLAY”键使磁带机启动，然后敲键盘上的回车键。稍等片刻（约 15 秒左右），屏幕显示“IT′ S LOADING PLEASE WAIT”（请稍候，正在装入）。当游戏程序装入内存后自动运行。此时，按下磁带机的“STOP”键。

PLOT X，Y

在低分辨率作图方式下，在（X，Y）位置上画一个点，其颜色由最近执行的 COLOR 确定。

POKE X，Y

将十进制数 Y 保存到 X 所示的内存单元中。
$-65535 \leqslant X \leqslant 65535, 0 \leqslant Y \leqslant 255$。

POP

作用与 RETURN 类似。区别在于：当执行 RETURN 时，堆栈顶部的地址被弹出，使计算机按照弹出的地址去执行。当执行 POP 时，栈顶地址也被弹出但计算机并不按弹出的地址去执行，而是执行 POP 语句的下一个语句。因此，在 POP 之后跟上一个 GOTO 语句，就可以使控制转至任何语句去执行。

PRINT [〈**表达式**〉] [{，〈**表达式**〉}…]

打印表达式的值。表达式之后可以跟逗号“，”或分号“；”。在敲键时也可用“?”来代替 PRINT。

READ **变量名** [，{**变量名**}…]

将 DATA 提供在数据区中的数据依次送到变量名中。

RESTORE

将数据区（由 DATA 提供的数据）中指针拨到第一个数据位置，使数据区中数据可以重新使用。

RETURN

子程序的结束标志。它将堆栈中保存的地址弹出，使计算机按照弹出的地址去执行。一个子程序中至少有一个 RETURN 语

句。子程序至多可以嵌套 25 层。

ROT＝m

设置向量绘图图形的旋转角度。

SCALE＝n

确定向量图形的比例大小。

SPEED＝〈**算术表达式**〉

改变计算机到 I/O 设备之间字符的传输速度。表达式的值确定了传输速度的大小，最低速度为 0，最高速度为 255，即算术表达式的值在 0～255 之间。

TEXT

由低分辨率图形方式或高分辨率图形方式转换到满屏文本显示方式。

TRACE

跟踪程序的执行，并将行号显示在屏幕上。用 NOTRACE 可以取消跟踪。

VTAB (〈**算术表达式**〉)

将光标移到指定的行上。注意：VTAB 只能使光标上下移动，而不能左右移动。在西文 BASIC 下，算术表达式的值必须在 1～24。在中文 BASIC 下，其值在 1～10 之间。

VLIN Y1，Y2 AT X

在低分辨率作图方式下以(X1，Y2)为终点画一条垂直线。

WAIT X，Y

使程序在执行过程中处于等待状态，直到指定单元的值满足一定条件时，程序又继续执行。其中：X 值表示存储单元地址，－65535≤X≤65535。Y 表示 0～255 之间的一个十进制数。操作过程是：

程序先读入 X 单元的内容再与 Y 值进行“与”运算，如果结

果为零则重复这一过程。若结果不为零，则程序向下继续执行。X，Y 也可以是算术表达式。

WAIT X，Z，Y

作用同 WAIT X，Y。X 表示存储单元地址，$-65535 \leqslant X \leqslant 65535$。Y，Z 为 0～255 之间的十进制数。操作过程是：

程序读 X 的值与 Z 值进行“异或”运算，然后再与 Y 值进行“与”运算，结果为零则重复这个过程，结果不为零则向下继续执行。

XDRAW n [AT X，Y]

作用同 DRAW，只不过图形颜色是 DRAW 命令的补色。这通常用来擦除一个图形。

1.2 BASIC 函数一览表

ABS(X)

求 X 的绝对值。X 可以是算术表达式。

ASC(〈**字符串表达式**〉)

求字符串表达式值中第一个字符的 ASCII 码值。字符串的长度不超过 255。

ATN(X)

求 X 的反正切值 arctgx。$-\pi/2 \leqslant X \leqslant \pi/2$。

CHR $ (〈**算术表达式**〉)

求出与算术表达式值相对应的 ASCII 码字符。算术表达式的值必须在 0～255 之间。

COS(X)

求 X 的余弦值 COSX。X 只能是弧度。

EXP(〈**算术表达式**〉)

求 e^x 的值。X 是算术表达式。

INT(X)

求最接近 X 的而又不大于 X 的最大整数。

LEFT $ (A $,n)

求出 A $ 中最左边的 n 个字符。A $ 可以是字符串表达式。

LEN(A $)

求 A $ 中字符的个数,即长度。

LOG(X)

求 X 的自然对数值 Lg(x)。x>0。

MID $ (A $,n[,m])

取出字符串 A $ 值的中间子串。n 表示开始字符位置,m 表示所取子串的字符个数。若省略 m,则取第 n 个字符开始后的全部字符作为子串。1≤n,m≤255。若 n 大于 A $ 中字符的个数,那么得到一个空串。

PDL(n)

这个函数用来读取#n 游戏控制器摇杆位置量的编码值(0~255)。0≤n≤3。

PEEK(X)

读取由 X 表示的内存单元的内容。-65535≤X≤65535。X 也可以是一个算术表达式。

POS (〈表达式〉)

求出当前光标所在的列位置。表达式的值不起任何作用,因而可简单的写成 0。

RIGHT $ (A $,n)

这个函数的作用是取出字符串右边的 n 个字符:

RND(X)

这是一个随机函数，其产生随机数的规则是：

X>0，产生 0～0.999999999 之间的一个随机数；

X=0，产生与上次相同的一个随机数；

X<0，产生与 X 值相对应的一个随机数。

SGN(X)

这是一个符号函数，其规则是：

$$SGN(X)=\begin{cases}1, X>0\\0, X=0\\-1, X<0\end{cases}$$

SIN(X)

求 X 的正弦值。X 只能是弧度值。

SORN (X,Y)

这个函数给出低分辨率图形方式下(X,Y)座标点的颜色值。通常，0≤X≤39，0≤Y≤47。

SPC(〈**算术表达式**〉)

产生若干空格，空格个数由算术表达式的值来确定。

SQR(X)

求 X 值的平方根。X≥O。

STR $ (〈**算术表达式**〉)

将算术表达式的值转换成字符型。

TAB(〈**算术表达式**〉)

以当前行的第 1 列为准，将光标移到指定的列。算术表达式的值必须在 0～255 之间。注意：TAB 语句只能使光标右移而不能向左回退；TAB 只能放在 PRINT 后面使用。

TAN(X)

求 X 的正切值 tgx。X 必须是弧度值。

USR(X)

将算术表达式 X 的值送到机器语言子程序中进行处理。操作过程是：

①计算出 X 的值并保存在＄0A～＄0C 单元中。

②转到＄0A 单元去执行。在＄0A～＄0C 单元中存放一条 JMP ××××指令，其中××××是用户自已设置的处理＄9D～＄A3 单元中 X 值的子程序入口。

③在机器语言执行完毕，即到 RTS 后该函数所得到的值将放在浮点累加器中。

1.3 BASIC 出错信息一览表

BAD SUBSCRIPT ERROR (107)

使用数组元素时下标不正确，或下标值超界。

CAN'T CONTINUE ERROR (255)

企图执行一个不存在的程序，或者在更正错误之后想继续执行程序。

DIVISION BY ZERO ERROR (133)

用 0 做除数。

FORMULA TOO COMPLEX ERROR (191)

语句形式太复杂。如果 IF 〈字符串〉 THEN 〈语句〉形式的语句嵌套了两层以上，就可能发生这类错误。

ILLEGAL DIRECT ERROR (254)

不能把 INPUT、DEF FN 、GET 或 DATA 语句作为立即方式来使用，而必须在程序中使用。

ILLEGAL QUANTITY ERROR (53)

非法量。字符串函数、数值函数的参数值超过范围。

NEXT WITHOUT FOR ERROR (0)

FOR 和 NEXT 不匹配。

OUT OF DATA ERROR (42)

缺少数据。执行 READ 时,DATA 提供的数据已经用完,没有数据可读。

OUT OF MEMORY ERROR (77)

内存不足。程序太大;变量太多;循环的嵌套超过 10 层;子程序的嵌套超过 24 层;括号嵌套超过 36 层;设置的 HIMEM:值太低;LOMEM:值太高等等都会发生这类错误。

OVERFLOW ERROR (69)

数据溢出错误。计算后的结果太大,超出了计算机所能表示的范围。

REDIM' D ARRAY ERROR (120)

重新定义数组出错。

RETURN WITHOUT GOSUB ERROR (22)

GOSUB 和 RETURN 不匹配。

STRING TOO LONG ERROR (176)

字符串太长,以致超过 255 个字符。

SYNTAX ERROR (16)

语法错误。保留字拼写不正确;括号不配对;一个语句行中夹杂了不应该有的字符。这些情况均可能发生这种错误。

TYPE MISMATCH ERROR (163)

数据类型不匹配。

UNDEF' D FUNCTION ERROR (224)

调用了未定义的函数。

UNDEF' D STATEMENT ERROR (90)

在 GOTO、GOSUB 或 THEN 中使用了不存在的语句行号。

附录二　LOGO 语言一览表

这里给出了本书中用到的 LOGO 的命令、控制以及常见的出错信息。对于固化 LOGO 与 APPLE－LOGO 之间的差别，也在表中注明，固化 LOGO 中没有的命令，在对应位置上用“无”字标出。

2.1 APPLE－LOGO 和固化 LOGO 命令对照表

APPLE－LOGO	用途（用法）	固化 LOGO
ARCTAN :A/:B	反正切函数	ATAN :A :B
BACK/BK 〈X〉	后退 X 步	
BUTFIRST/BF 〈字/表〉	去掉前一字符	
BUTLAST/BL 〈字/表〉	去掉后一字符	
CATALOG	显示磁盘文件目录	
CHAR 17	打印图形	
CLEAN	清屏，不复位	CS
CLEARSCREEN/CS	清屏，复位	DRAW
CONTINUE/CO	取消暂停，恢复运行	
COS 〈X〉	余弦函数	
. DEPOSIT 1913 〈X〉	设置打印格式	
DOT [X Y]	打点命令	无
END	定义过程结束标志	
EDIT/ED "〈过程名〉	进入编辑状态	ED〈过程名〉
ER [〈过程名 A〉〈过程名 B〉…]	清除部分过程	无
ERALL	清除内存全部过程	ER ALL
ERASE/ER "〈过程名〉	清除指定过程	ER〈过程名〉
ERASEFILE "〈文件名〉	删除磁盘指定文件	

APPLE－LOGO	用途（用法）	固化 LOGO
FIRST〈字/表〉	取前一字符	
FORWARD/FD〈X〉	前进 X 步	
FPUT〈字/表〉〈表〉	前〈字/表〉置后表前	
HEADING	测定海龟方向	
HIDETURTLE/HT	海龟消隐	
HOME	不清屏，复位	
IF〈条件〉[命令序列]	满足条件执行[命令序列]	IF〈条件〉〈命令〉
IF AND〈条件 1〉〈条件 2〉[……]	“与”逻辑条件表达式	FI ALLOF（…）（…）〈…〉
IF OR〈条件 1〉〈条件 2〉[……]	“或”逻辑条件表达式	IF ANYOF（…）（…）〈…〉
IF NOT〈条件〉[命令]	“非”逻辑条件表达式	IF NOT〈条件〉〈…〉
INT〈变量〉	舍小数取整	INTEGER
LAST〈字/表〉	取后一个字符	
LEFT/LT〈X〉	海龟左转 X 度	
LOAD "〈文件名〉	将文件读入内存	READ "〈文件名〉
LPUT〈字/表〉〈表〉	将〈字/表〉置于表后	
MAKE "〈变量名〉〈表达式〉	表达式值赋给变量	
OUTPUT/OP〈变量/表达式〉	传送〈…/…〉值	
PENDOWN/PD	落笔	
PENUP/PU	抬笔	
PO "〈过程名〉	调阅（打印）过程	PO〈过程名〉
POALL	调阅（打印）全部过程	POALL
POTS	调阅（打印）过程名	
PR（〈过程名〉〈参数…〉）	打印过程结果	
PRINT/PR〈变量/表达式〉	打印〈…/…〉值	
. PRINTER 1/0	联接/断开打印机	OUTDEV 1/0
PRODUCT〈X〉〈Y〉	求积（X·Y）函数	无
QUOTIENT〈X〉〈Y〉	商（X/Y）取整函数	
RANDOM〈X〉	随机函数 0～（X－1）	
REMAINDER〈X〉〈Y〉	取余（X/Y）函数	
REPEAT〈X〉[命令序列]	将□中命令重复 X 次	
RIGHT/RT〈X〉	海龟右转 X 度	
ROUND〈X〉	舍入取整函数	

APPLE—LOGO	用途（用法）	固化 LOGO
SAVE "〈文件名〉	将内存文件存盘	
SENTENCE/SE〈字/表/数〉	连接字/表/数	
SETBG〈X〉	设置屏幕底色	BG〈X〉
SETPOS [XY]	坐标绘图命令（实参）	SETXY〈X〉〈Y〉
SETPOS SE :X :Y	坐标绘图命令（形参）	同上
SETHEADING/SETH〈X〉	海龟顺时针转 X 度	
SETPC〈X〉	设置画笔颜色	PC〈X〉
SETSCRUNCH〈X〉	纵横比例设置	. ASPECT〈X〉
SETX〈X〉	海龟水平移 X 处	
SETY〈Y〉	海龟垂直移至 Y 处	
SHOWTURTLE/ST	显示海龟	
SIN〈X〉	正弦函数	
SQRT〈X〉	平方根函数	
STOP	停止过程	
SUM〈X〉〈Y〉	求（X+Y）的和函数	无
TO〈过程名〉	过程定义标志	
TYPE〈变量〉	紧凑打印	PRINT1〈变量〉
WAIT〈T〉	延时	无
WORD〈字 1〉〈字 2〉	连接字	无
XCOR	测定海龟 X 位置	
YCOR	测定海龟 Y 位置	

2.2 控制键

1. 直接命令下使用的控制键

APPLE—LOGO	作用	固化 LOGO
CTRL+T	全文本显示	
CTRL+S	图形/文本混显	
CTRL+L	全图形显示	CTRL+F
CTRL+Y	重显前次命令	CTRL+P
CTRL+W	暂停（用任意键恢复）	
CTRL+Z	暂停（用 CO 恢复）	

2. 编辑状态下使用的控制键

CTRL+P	光标上移	
CTRL+N	光标下移	
CTRL+O	光标处产生空行	
CTRL+C	保存修改，退出编辑	
CTRL+G	撤销修改，退出编辑	

3. 直接与编辑状态下使用的控制键

CTRL+A	光标调至行首	
CTRL+E	光标调至行尾	
CTRL+K	删除光标右侧字符	△键
CTRL+D	删除光标处字符	
CTRL+B	光标左移不清字符	◁键
◁键	光标左移，清左侧字符	ESC 键
▷键	光标右移，不清字符	

2.3 LOGO 常见出错信息表

1. 〈A〉CAN′ T BE USED IN PROCEDURE IN〈B〉。
   A 不能用在过程 B 中。
2. CAN′ T DIVIDE BY ZERO
   零不能作除数。
3. 〈+〉DOSEN′ T LIKE AS INPUT IN〈A〉
   “+”符号不能作为参数输入到 A 中。
4. 〈过程〉DIDN′ T OUTPUT TO〈命令〉。
   过程没有输出任何参数以供命令使用。
5. FILE ALREADY EXISTS。
   文件已经存在。即用户文件同先前的文件同名了。
6. FILE NOT FOUND。
   文件没有找到。即敲入的文件名不在磁盘上。

7. 〈FD〉 IS A PRIMITIVE〈…〉。
   “FD”是基本命令…。
8. 〈A〉HAS NO VALUE IN〈B〉。
   A没有值送给B。
9. I DON′ T KNOW HOW TO〈…〉。
   我不认识〈…〉是什么。
10. I DON′ T KNOW WHAT TO DO WITH〈数据〉。
    我不知道数据用来干什么。
11. 〈过程〉IS ALREADY DEFINED。
    过程已经定义。
12. NOT ENOUGH INPUT TO〈过程〉。
    调用过程时没有足够的输入项。
13. NUMBER TOO BIG。
    数值太大。
14. OUT OF SPACE IN 。
    内存空间已用完。
15. TURTLE OUT OF BOUNDS。
    海龟超界。



封面设计：阎欢玲

ISBN 7-5053-1046-1/TP·169　定价：4.40 元


BASIC与Logo语言
電子工業出版社